JOÃO BOSCO A. PEREIRA FILHO

UM ALGORITMO DE FILTRAGEM COLABORATIVA BASEADO EM SVD

FLORIANÓPOLIS

2010

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

UM ALGORITMO DE FILTRAGEM COLABORATIVA BASEADO EM SVD

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Ciências da Computação

JOÃO BOSCO A. PEREIRA FILHO

Florianópolis, Novembro de 2010

UM ALGORITMO DE FILTRAGEM COLABORATIVA BASEADO EM SVD

JOÃO BOSCO A. PEREIRA FILHO

Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Ciências da Computação e aprovada em sua forma final pelo Programa de Pós Graduação em Ciências da Computação da Universidade Federal de Santa Catarina.

_____

Renato Fileto, Dr.
Orientador

_____

Mário Dantas, Dr.
Coordenador do Programa de Pós-Graduação em Ciências da Computação

Banca Examinadora:

_____

Renato Fileto, Dr.
Presidente

_____

Pedro Alberto Barbetta, Dr.

_____

Jerusa Marchi, Dra.

_____

Roberto Willrich, Dr.

DEDICATÓRIA

Ao Bosco pai, é uma grande responsabilidade ser seu filho ...

AGRADECIMENTOS

Este trabalho não poderia ter sido concluído sem a contribuição de algumas pessoas. Agradeço primeiramente à minha família: Lais, Ligia, Fátima e Bosco por seu amor, apoio e compreensão. Agradeço a minha namorada Joice, por todo seu amor e por sua revisão da redação da dissertação.

Agradeço também a todos os integrantes da Chaordic Systems, empresa que nasceu graças a trabalhos acadêmicos como este: Leopoldo, Fausto, Ivando, Alceu, Rossato e, em especial, João Lourenço, meu sócio, que sempre me apoiou e estimulou a continuar com o trabalho de mestrado. Agradeço também aos novos sócios da empresa Paulo Caputo e Jorge Steffans, que acreditaram no nosso sonho de transformar o conhecimento acadêmico em um produto de mercado. Agradeço à Fundação CERTI, em especial, Carlos Alberto Schneider e Ricardo Teixeira, que nos apoiaram no princípio desta empreitada.

Um agradecimento à Verinha, secretária do PPGCC-UFSC, sempre bem-humorada e disposta a resolver os problemas que os alunos lhe apresentavam. Um agradecimento também aos professores: Mário Dantas, que me convenceu a mudar de tema e continuar o trabalho acadêmico, Renato Fileto, que me acolheu no PPGCC e me ajudou a combater todas as adversidades que surgiram no decorrer deste trabalho, e ao professor Pedro Barbetta, por sua co-orientação e suas aulas de Estatística. Sem a ajuda do professor Barbetta, este trabalho nunca teria evoluído. Um agradecimento especial ao professor Guilherme Bittencourt (*in memoriam*), não só pelo apoio dado a este trabalho através de suas explicações sempre muito didáticas e precisas, mas principalmente pelo exemplo de pessoa que ele foi. Esta foi sem dúvida sua melhor aula.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências da Computação.

UM ALGORITMO DE FILTRAGEM COLABORATIVA BASEADO EM SVD

JOÃO BOSCO A. PEREIRA FILHO

Novembro / 2010

Orientador: Renato Fileto, Dr..
Área de Concentração: Ciências da Computação.
Palavras-chave: sistemas de recomendação, SVD, filtragem colaborativa, PCA, KNN, banco de dados.
Número de Páginas: 82

A presente dissertação tem como objetivo contribuir com a pesquisa na área de sistemas de recomendação e propõe um algoritmo de filtragem colaborativa baseado em Decomposição por Valor Singular (SVD) que modela o perfil de um grande grupo de usuários, com o intuito de fazer recomendações personalizadas a eles. Tal algoritmo utiliza técnicas normalmente utilizadas no treinamento de redes neurais artificais, técnicas de estatística e de álgebra linear para processar as recomendações. É proposta uma maneira inteligente de se inicializar o modelo do algoritmo, que acaba por acelerar a convergência do treinamento e melhora a eficácia do mesmo. As experimentações foram realizadas no contexto do concurso *Netflix Prize*, que disponibiliza uma grande base de dados e uma metodologia de avaliação dos resultados. Tais experimentações realizadas e uma análise comparativa com outro algoritmo demonstram que o algoritmo proposto retorna resultados mais precisos, apesar de ser mais lento. Além disso, o algoritmo também foi experimentado em conjunto com este outro algoritmo e as recomendações geradas por esta combinação se mostraram ainda mais precisas.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

AN SVD BASED ALGORITHM TO COLLABORATIVE FILTERING

JOÃO BOSCO A. PEREIRA FILHO

Nov / 2010

Advisor: Renato Fileto, Dr..
Area of Concentration: Area de concentração em inglês.
Keywords: recommender systems, SVD, collaborative filtering, PCA, KNN, databases.
Number of pages: 82

This dissertation aims to contribute to research in the area of recommendation systems and proposes a collaborative filtering algorithm based on Singular Value Decomposition (*SVD*) that models the profiles of a large number of users, in order to make personalized recommendations for them. It uses techniques commonly used in artificial neural networks training, statistical techniques and linear algebra to handle the recommendations. It is presented a clever way to initialize the algorithm model, which turns out to accelerate the convergence of training and improves efficiency. The experiments were conducted in the context of the competition *Netflix Prize*, which provides a large database and a methodology for evaluating the results. Such experiments and a comparative analysis with other algorithm demonstrates that the algorithm returns more accurate results, despite of being slower. Moreover, the algorithm was also tested in conjunction with this other algorithm and the recommendations generated by this combination proved to be even more accurate.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

SVD	Decomposição por Valor Singular, técnica algébrica de fatoração de matrizes
PCA	Análise de Componentes Principais, técnica de análise estatística multivariada
KNN	K Nearest Neighbours, algoritmo de aprendizagem de máquina baseado na semelhança entre vizinhos
SBC	Sistemas Baseados em Conteúdo, tipo de sistema de recomendações que utiliza características dos itens para fazer a recomendação
SFC	Sistemas de Filtragem Colaborativa, tipo de sistema de recomendaçãoo que utiliza a interação dos usuários com o sistema para fazer a recomendação
RMSE	Raiz Quadrada do Erro Médio Quadrático, medida utilizada para avaliar erro em predições

1 INTRODUÇÃO

> *If money is your hope for independence you will never have it. The only real security that a man will have in this world is a reserve of knowledge, experience, and ability.*
>
> — Henry Ford

A facilidade de criação e publicação de conteúdos na Internet e a crescente quantidade de informações disponíveis fazem com que cada novo item colocado na Internet dispute o recurso mais escasso da sociedade digital: a atenção dos usuários. Em resposta aos desafios da sobrecarga de informação e a necessidade de personalização na Internet, tem-se procurado desenvolver sistemas nos quais os usuários possam rapidamente identificar os conteúdos que mais lhes interessam. A dificuldade é ainda maior quando o gosto do usuário é fator fundamental para se conhecer a relevância de uma determinada informação. Para saber quais são os itens mais interessantes, proveitosos, valiosos ou divertidos para uma determinada pessoa, é fundamental entender seu gosto.

Os sistemas de recomendação têm por objetivo facilitar o acesso de usuários às informações que mais lhes interessam. Para tanto, tais sistemas tentam descobrir o gosto de seus usuários e então recomendar a estes, novos itens que provavelmente irão agradá-los. Estes sistemas são bastante comuns em sites de comércio eletrônico, como o sistema de recomendação utilizado pela empresa Amazon.com, que é responsável por 35% das vendas de produtos da empresa (MARSHAL, 2006).

Outro exemplo é a empresa *Netflix*, uma das maiores locadoras de filmes dos Estados Unidos, que lançou em 2006 um desafio chamado *Netflix Prize* (NETFLIX, 2006), oferecendo um prêmio de um milhão de dólares para quem conseguir uma melhoria de 10% no nível de precisão do seu atual sistema de recomendação de filmes para clientes (BENNETT et al., 2007). Milhares de equipes participantes, espalhadas por todo o mundo, propuseram novas e melhores soluções, sendo a equipe *BellKor's Pragmatic Chaos* declarada a campeã em Setembro de 2009 (NETFLIX, 2009).

Os sistemas de recomendação podem ser subdivididos em duas categorias: baseados em conteúdo e baseados em filtragem colaborativa. Os siste-

mas baseados em conteúdo utilizam as informações sobre os itens disponíveis para fazer uma recomendação. Por exemplo, um sistema de recomendação de artigos científicos pode usar as palavras-chave dos artigos para fazer as recomendações (TORRES, 2004).

Os sistemas baseados em filtragem colaborativa, por outro lado, fazem recomendações a um usuário com base nas ações de outros usuários. Um algoritmo típico de filtragem colaborativa pode, por exemplo, avaliar que João tem gostos similares aos de Maria e, então, recomendar para ele os itens que Maria gosta. A combinação de comportamento, preferências ou idéias de um grupo de pessoas para a criação de novos conhecimentos é chamada inteligência coletiva (SEGARAN, 2007).

Existem diversas técnicas que podem ser utilizadas em filtragem colaborativa. A Análise de Componentes Principais (*PCA*), por exemplo, é uma técnica de análise estatística multivariada que, quando utilizada em filtragem colaborativa, tenta identificar características relevantes nos dados analisados. Estas características podem ser interpretadas como o gosto dos usuários e as características dos itens a serem recomendados. Tais padrões identificados são utilizados, então, para se recomendar os itens que provavelmente irão agradar mais os usuários do sistema.

Existem também diversas formas de se fazer PCA (R, 2010a). Uma das técnicas mais conhecidas é a Decomposição por Valor Singular (SVD). A SVD é uma técnica algébrica de fatoração de matrizes e pode ser utilizada em filtragem colaborativa para se fazer PCA, ou seja, descobrir características latentes dos usuários e dos itens que estão sendo avaliados. Desta maneira, pode-se inferir o gosto de cada indivíduo e, então, fazer recomendações. Existem diversas implementações de SVD disponíveis em produtos, como o *Matlab* (MATLAB, 2010) ou o *R Project* (R, 2010b), por exemplo. Entretanto, a aplicação direta de algoritmos de SVD aos problemas de filtragem colaborativa não garante boa eficácia, sendo necessários diversos ajustes para desenvolver bons algoritmos (MA, 2008). Além do problema de eficácia gerado por estas ferramentas, as implementações disponíveis destas não conseguem processar um grande volume de dados e são, portanto, inadequadas para o uso em filtragem colaborativa de grandes bases de dados.

1.1 Objetivos

Este trabalho tem como objetivo geral desenvolver e validar um algoritmo de recomendação baseado em filtragem colaborativa para grandes bases de dados utilizando SVD. Para tanto, são utilizadas técnicas de treinamento

de redes neurais artificais e de tratamento estatístico dos dados analisados. Tal proposta é fundamentada em diversos trabalhos presentes na literatura e inclui também inovação.

O algoritmo proposto visa atender sistemas de recomendação que lidam com grandes volumes de dados e oferecem suporte a avaliações, isto é, sistemas em que usuários podem avaliar explicitamente os diversos itens disponíveis. O algoritmo analisa as avaliações realizadas e, então, faz a predição das notas que usuários dariam a itens ainda não avaliados. Tais predições poderiam ser utilizadas para se fazer as recomendações de maneira personalizada aos usuários do sistema. Como normalmente são poucos os usuários que se dispõem a avaliar itens e as matrizes de avaliações são muito grandes, o algoritmo proposto neste trabalho também trata o problema da esparsidade destas matrizes.

Foi atribuída uma ênfase à precisão do algoritmo proposto. Ou seja, este trabalho tem como objetivo apresentar um algoritmo baseado em SVD capaz de lidar com grandes bases de dados e que gere boas predições de avaliações, independente do tempo de processamento necessário para execução do mesmo. Boas predições são obtidas quando a taxa de erro obtida na predição é pequena.

Os objetivos específicos deste trabalho incluem:

- Efetuar pesquisa bibliográfica sobre sistemas de recomendação, com ênfase em filtragem colaborativa e nas técnicas de Decomposição por Valores Singulares (SVD), Análise de Componentes Principais (PCA) e K Nearest Neighbours (*KNN*).

- Desenvolver um algoritmo de recomendação baseado em filtragem colaborativa utilizando SVD, que deve ser capaz de lidar com matrizes de avaliações grandes e esparsas para inferir a recomendação de itens não avaliados.

- Implementar o algoritmo desenvolvido de maneira que o mesmo possa ser executado com uma grande base de dados de avaliações que usuários atribuem à itens.

- Realizar experimentos com o algoritmo nesta base de dados e fazer comparações da eficácia do mesmo com outra solução.

- Combinar o algoritmo proposto com outra solução afim de obter um algoritmo resultante mais eficáz que os anteriores.

1.2 Justificativa

Os experimentos descritos neste trabalho foram realizados no contexto do concurso Netflix Prize (BENNETT et al., 2007), que fornece uma grande base de dados para experimentações, além de uma metodologia de avaliação da eficácia dos algoritmos.

Sendo a base de dados do Netflix Prize uma das maiores bases disponíveis livremente para pesquisa em sistemas de recomendação, a escolha da mesma foi de crucial importância para o desenvolvimento deste trabalho, já que algoritmos de filtragem colaborativa precisam de um grande volume de dados.

Além disso, a metodologia de avaliação do concurso possibilitou o foco do trabalho no desenvolvimento e melhoria do algoritmo proposto, não havendo a necessidade de pesquisar sobre métodos de avaliação para o mesmo.

Apesar de ter sido experimentado com a base de dados do Netflix Prize, o algoritmo proposto é suficientemente genérico para poder ser utilizado em outros contextos.

Foi escolhida a técnica SVD, pois esta se mostrou muito eficiente no concurso (FUNK, 2006) (YU et al., 2009) (BELL; KOREN; VOLINSKY, 2007). Além disso, a técnica ainda foi pouco explorada no contexto de filtragem colaborativa, havendo muitas oportunidades de pesquisa no tema.

1.3 Limitações do trabalho

O algoritmo proposto faz apenas predições das notas que usuários atribuiríam a itens ainda não avaliados e não trata da recomendação em si. Entretanto, a recomendação poderia ser realizada de maneira trivial, através da escolha dos itens para serem recomendados com maior nota predita para o usuário em questão, por exemplo.

O algoritmo apresentado neste trabalho foi elaborado para ser treinado de maneira off-line, não sendo adequado para prover recomendações imediatas. Algoritmos com treinamento on-line devem ser elaborados de maneira a prover recomendações muito mais rapidamente.

Este trabalho também não aborda a atualização do modelo de recomendação, o que poderia, por exemplo, identificar mudanças nos gostos dos usuários. O fator tempo não é levado em consideração pelo algoritmo proposto.

Por fim, o algoritmo utiliza somente as avaliações dos usuários, não

utilizando outras informações também fornecidas no Netflix Prize, como os títulos dos filmes e as datas das avaliações. Não são utilizadas também quaisquer outras informações sobre os filmes que possam estar disponíveis em outras bases de dados.

1.4 Estrutura da dissertação

Este trabalho está dividido em seis capítulos, descritos a seguir.

O Capítulo 2 apresenta mais detalhadamente a área de pesquisa em sistemas de recomendação, atribuindo um enfoque maior à filtragem colaborativa.

O Capítulo 3 apresenta as técnicas de Decomposição por Valores Singulares (SVD), Análise de Componentes Principais (PCA) e suas relações com filtragem colaborativa.

O Capítulo 4 descreve o algoritmo proposto, esclarecendo cada decisão de seu projeto e os pontos de inovação, além de fazer alguns esclarecimentos com relação à implementação.

O Capítulo 5 relata os experimentos realizados com o algoritmo SVD e faz uma análise comparativa com um algoritmo baseado em KNN, técnica muito utilizada em filtragem colaborativa e apresentada no mesmo capítulo. Além disso, são esclarecidas as regras e forma de validação de algoritmos utilizada no Netflix Prize, cuja base de dados é utilizada na validação do algoritmo proposto neste trabalho.

Por fim, o Capítulo 6 apresenta as conclusões, as possibilidades de trabalhos futuros e os trabalhos relacionados.

2 SISTEMAS DE RECOMENDAÇÃO

E no fim não são os anos da sua vida que contam, mas sim a vida em seus anos.

— Abraham Lincoln

A segunda metade do século XX foi marcada pela cultura de massa, na qual os filmes, livros e músicas mais vendidos eram o foco de atenção do mercado. Isto ocorreu porque os meios de divulgação, como a televisão, por exemplo, eram caros. Portanto fazia sentido divulgar somente os produtos que mais vendiam. Além disso, o espaço físico em livrarias tem um custo relativamente alto, forçando os vendedores a oferecer apenas os produtos que são campeões de vendas. Este fenômeno pode ser explicado pela regra do 80/20, que significa que 80% do lucro são obtidos com a venda de apenas 20% dos produtos (ANDERSON, 2006).

Com o advento e a popularização da Internet nos anos 90, estas regras de mercado começaram a mudar. A internet possibilitou o oferecimento de uma infinidade de produtos, já que o espaço virtual ocupado por um produto é muito mais barato do que um espaço na prateleira. Além disso, a Internet pode possibilitar um alcance de clientes muito maior do que o alcançado em uma loja tradicional.

Surgiu, então, o fenômeno da "cauda longa", primeiramente observado pelo pesquisador e editor-chefe da revista Wired, Chris Anderson (ANDERSON, 2006). O pesquisador analisou estatísticas de vendas de lojas de comércio eletrônico e constatou que a receita obtida pelas vendas dos produtos menos populares supera a receita obtida com as vendas dos produtos mais populares. O termo "cauda longa" refere-se ao formato dos gráficos de vendas destas empresas, nos quais os produtos menos populares, apesar de venderem pouco, são muitos e se localizam na cauda destes gráficos. A Figura 1 apresenta o gráfico de vendas de 2003 da empresa Amazon, no qual o eixo vertical indica o total de vendas e o horizontal os produtos da empresa ordenados de forma decrescente de popularidade. Pode-se observar que o total de receita obtido pelos produtos menos populares, representado pela área cinza do gráfico, supera o total de receita obtido pelos produtos mais populares.

Anderson defende a estratégia de que é mais interessante explorar

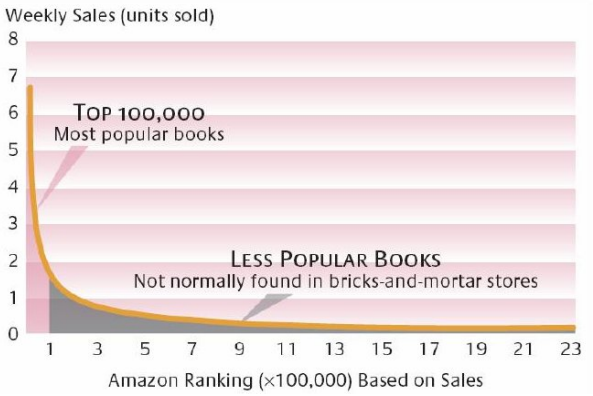

Figura 1: Vendas da empresa Amazon em 2003
(*Fonte: Management Science, Novembro 2003*)

o mercado de cauda longa do que o mercado de massa. Para tanto, é necessária a utilização de ferramentas adequadas, para que os usuários de lojas de comércio eletrônico possam encontrar produtos que lhes interessam com maior facilidade.

As ferramentas mais comumente utilizadas com este fim são as ferramentas de busca, nas quais os usuários informam palavras-chave relacionadas com o que se quer buscar e uma lista de opções é apresentada a eles, ou seja, os usuários têm um papel ativo na recuperação de informação.

Sistemas de recomendação, por outro lado, são ferramentas que podem ser utilizadas passivamente pelos usuários para explorar as possibilidades oferecidas pelo mercado, não havendo a necessidade de qualquer ação explícita dos mesmos para receber as recomendações. Estes sistemas analisam o comportamento de usuários para tentar entender o gosto dos mesmos, analisam os itens disponíveis e recomendam a estes usuários itens que provavelmente irão agradá-los. Tais sistemas podem, portanto, ajudar os usuários a encontrar produtos de nicho específico e melhorar as vendas de lojas virtuais.

O primeiro sistema de recomendação de que se tem notícia é o sistema Tapestry (GOLDBERG et al., 1992), desenvolvido por pesquisadores do laboratório PARC (Palo Alto Research Center, Xerox). Tal sistema foi desenvolvido na década de 90 com o objetivo de facilitar a filtragem relevante de emails utilizando a ajuda humana, já que os usuários do sistema podiam fazer anotações sobre as mensagens.

2.1 Definição formal do problema de recomendação

O problema a ser resolvido por sistemas de recomendação pode ser definido formalmente da seguinte maneira (BERNARTT, 2008):

Definição 2.1.1 *: Seja* $\mathbf{M}$ *o conjunto de todos os itens disponíveis em um sistema,* $\mathbf{U}$ *o conjunto de todos os usuários do sistema,* $f : \mathbf{M} \times \mathbf{U} \longrightarrow R$ *a função de utilidade que mede o quanto um determinado item* $m \in \mathbf{M}$ *é útil a um usuário* $u \in \mathbf{U}$*, sendo R um conjunto completamente ordenado. Procura-se obter um item* m'_u *que maximiza a função de utilidade para o usuário u. Ou seja, procura-se obter o item mais útil ao usuário em questão. Tal definição pode ser resumida através da seguinte equação:*

$$\forall u \in \mathbf{U}, \quad m'_u = \arg\max_{m \in \mathbf{M}} f(u, m) \tag{2.1}$$

Os sistemas de recomendação podem ser classificados em dois grupos: sistemas baseados em conteúdo (*SBC*) e sistemas de filtragem colaborativa (*SFC*). SBCs se baseiam nas características dos itens para fazer a recomendação, enquanto que SFCs se baseiam nas relações entre usuários e itens, como as notas que usuários atribuem a itens.

2.2 Sistemas de recomendação baseados em conteúdo (SBC)

SBCs utilizam informações sobre os próprios itens disponíveis para fazer as recomendações e, por isto, são dependentes do domínio de aplicação. Chen e Chen (2005), por exemplo, propõe um sistema de recomendação de músicas que analisa o próprio conteúdo da música para agrupar as músicas parecidas e então recomendar aos usuários.

Neste tipo de sistema, normalmente os usuários precisam preencher um perfil inicial para indicar ao sistema o seu gosto pessoal. Desta forma, os usuários destes sistemas possuem perfis e os itens disponíveis possuem descrições. Os perfis dos usuários são confrontados com os itens, e aqueles que possuem maior similaridade com o perfil de um determinado usuário são recomendados.

Estes sistemas têm sua origem na área de recuperação de informações textuais e utilizam diversas técnicas desta área. É comum representar tanto os perfis dos usuários quanto os itens disponíveis de forma textual e utilizar técnicas de similaridade entre documentos para se determinar quais itens são mais alinhados às preferências de um determinado usuário. Desta forma,

os itens mais próximos das preferências do usuário em questão devem ser recomendados a ele.

Torres et al. (2004) propõe um sistema de recomendação de artigos científicos, no qual o perfil do usuário é modelado utilizando-se outros artigos que o usuário tenha gostado. Os artigos do perfil de um usuário são comparados com novos artigos disponíveis e caso haja semelhança entre eles, estes serão recomendados.

Um método comumente utilizado para medir semelhança é a comparação dos pesos dos termos contidos nos documentos. A medida de peso mais utilizada para isto é a Term Frequency - Inverse Document Frequency (TI-IDF) (SALTON; BUCKLEY, 1987). Esta comparação é similar à comparação de palavras-chave com documentos em sistemas de recuperação de informação textual (BAEZA-YATES; RIBEIRO-NETO, 2008), evidenciando a estreita relação que existe entre SBCs e esta área.

O TF-IDF associa a cada palavra um valor que cresce proporcionalmente a medida que aumenta a frequência desta palavra no texto, mas diminui com o aumento da frequência desta palavra no conjunto de todos os textos. Desta forma, palavras que são muito comuns em um conjunto de textos não são adequadas para descrever tais textos. Por outro lado, palavras frequêntes em um texto, mas raras em outros textos, são boas candidatas a sintetizar o mesmo e diferenciá-lo de outros.

2.3 Sistemas de filtragem colaborativa (SFC)

SFCs utilizam algumas informações sobre a interação de seus próprios usuários para prover as recomendações, ou seja, as ações de um usuário no sistema podem influenciar nas recomendações para outros usuários. Este tipo de recomendação é bastante comum em lojas de comércio eletrônico, como a Amazon.com, que possui recomendações do tipo “quem comprou o produto X, também comprou Y”.

As bases de dados de sistemas de recomendação devem possuir informações sobre o histórico de ações de seus usuários, como os itens que foram buscados, acessados, avaliados, comprados e etc. Estas informações servem de feedback ao sistema, que as analisam para fazer as recomendações. Tais feedbacks podem ser explícitos ou implícitos. No primeiro caso, o usuário expressa diretamente o seu gosto, como por exemplo, quando um usuário atribui uma nota a um determinado produto. Já no segundo caso, o usuário expressa indiretamente o seu gosto, como no caso de uma visualização de um produto, por exemplo.

Por expressar diretamente o gosto do usuário, o feedback explícito é mais fácil de se utilizar para a geração das recomendações. Entretanto, nem sempre os usuários tem vontade de expressar o seu gosto. Além disso, os usuários são, muitas vezes, inconsistentes e podem acabar introduzindo ruído no sistema, o que dificulta o processamento de recomendações (AMATRIAIN; PUJOL; OLIVER, 2009). Alguns pesquisadores desenvolveram técnicas para estimular a obtenção de mais e melhores feedbacks explícitos. Pearl e Chen (2008) propõem algumas diretrizes para a elicitação, revisão e interface de explicação das preferências de usuários.

Já no caso dos feedbacks implícitos, muitas vezes o usuário sequer sabe que sua ação será utilizada para o processamento de recomendações. Entretanto, a confiança que se tem nesta informação é menor que a que se obtém no caso de um feedback explícito. Hu, Koren e Volinsky (2008) propõem uma estratégia de se representar o feedback implícito em duas magnitudes: preferência, que indica se os usuários gostaram ou não de um determinado item, e nível de confiança, que estima o quão confiante é a informação de preferência. Os pesquisadores desenvolveram um algoritmo para tratar deste paradigma e aplicaram o mesmo em um sistema de recomendação de programas de TV, onde o tempo que um usuário gasta para assistir um programa é levado em consideração para a geração de recomendações. Deste modo, se um usuário assistiu um programa por pouco tempo, por exemplo, não se tem muita confiança nesta informação e a mesma não influenciará nas recomendações que o usuário receber.

O objetivo principal dos SFCs é extrair o conhecimento coletivo existente nestas bases de dados através da detecção de padrões interessantes contidos nelas. Desta forma, é possível classificar os itens de acordo com a percepção humana de qualidade, o que é bastante complicado de fazer de forma algorítmica sem influência humana.

O SFC GroupLens (RESNICK et al., 1994) foi criado para ajudar os usuários de netnews, antigo sistema de discussão na Internet, a encontrar artigos que eles vão gostar em meio a uma quantidade muito grande de artigos disponíveis. Ele utiliza avaliações de usuários para os artigos e tenta prever a nota deles para artigos ainda não avaliados, auxiliando o usuário a escolher os artigos mais interessantes para ler. A predição para um determinado usuário é realizada através da seleção de usuários similares a ele, que avaliaram artigos de forma semelhante. As notas destes usuários para itens não avaliados pelo usuário em questão são então consideradas para fazer a predição. Além de ser um sistema de recomendação de artigos, o GroupLens foi elaborado para ser uma plataforma aberta de soluções de recomendação, podendo ser

utilizada em outros domínios se necessário. Esta independência do conteúdo sendo recomendado é na verdade uma característica de SFCs.

Outra característica interessante destes tipos de sistemas é que as recomendações geradas podem surpreender os usuários [1] (HERLOCKER et al., 2004). Ou seja, o usuário não precisa preencher um perfil informando, por exemplo, que gosta de filmes de ação para que o sistema recomende estes tipos de filmes para ele. A filtragem colaborativa pode detectar que provavelmente o usuário gostaria de um determinado filme de ação baseado na própria interação do usuário com o sistema. A detecção de padrões estatísticos complexos nos dados acaba por gerar recomendações não óbvias que podem surpreender os usuários. Esta é a principal característica de SFCs que ajuda na exploração de "produtos da cauda-longa", já que o objetivo é recomendar itens menos populares, mas que provavelmente o usuário irá gostar.

Alguns pesquisadores inclusive criaram métodos para explorar melhor a capacidade de SFCs surpreenderem os usuários ao fazerem recomendações de itens menos populares (SARWAR et al., 2001). Uma estratégia utilizada é dividir a probabilidade de um usuário gostar de um item pela probabilidade média dos usuários do sistema gostarem do mesmo item. Quanto maior o resultado desta razão, mais recomendável é o item ao usuário. Desta forma, são recomendados os itens que provavelmente irão agradar o usuário, mas que não são tão populares e óbvios.

SFCs podem utilizar as notas que os usuários atribuem a itens, as quais representam o grau de afinidade dos usuários com os mesmos. O objetivo é descobrir itens ainda não avaliados por usuários e que provavelmente irão agradá-los, ou seja, o problema de recomendação se reduz ao problema de predizer as notas que usuários atribuiriam a itens não avaliados. Notas altas atribuídas a itens significam que estes devem ser recomendados para o usuário em questão.

A Tabela 1 apresenta um exemplo de matriz de dados de avaliações de filmes. Neste exemplo, o usuário João atribuiu nota 5 ao filme O Poderoso Chefão, nota 1 a Miss Simpatia, não avaliou Duro de Matar e atribuiu nota 4 a Matrix. O objetivo de SFCs é completar esta matriz predizendo as avaliações desconhecidas. Nota-se que Lais tem preferências de filmes semelhantes as de Fátima, já que ambas atribuíram as mesmas notas para o filme Miss Simpatia, Duro de Matar e Matrix. Um sistema de filtragem colaborativa poderia, por exemplo, predizer que Lais atribuiría nota 2 ao filme Poderoso Chefão, já que foi esta a nota dada por Fátima ao mesmo filme.

[1] Serendipity (substantivo em inglês): Boa sorte para fazer descobertas inusitadas e afortunadas.

	O Poderoso Chefão	*Miss Simpatia*	*Duro de Matar*	*Matrix*
João	5	1	?	4
Lais	?	4	1	1
Lígia	?	5	?	1
Fátima	2	4	1	1

Tabela 1: Matriz de avaliações de filmes

Entretanto, a predição de avaliações em um sistema de filtragem colaborativa pode ser uma tarefa bastante difícil, já que em um sistema real esta matriz pode ser muito grande, devido ao grande número de usuários e itens, e ao mesmo tempo ser extremamente esparsa, pois são poucos os usuários que se dispõem a avaliar itens.

Um problema comum a todos os SFCs é o chamado problema do "início frio" (cold start) (MALTZ; EHRLICH, 1995). Tanto novos itens podem não ser recomendados enquanto nenhum usuário os avaliar, quanto novos usuários podem obter recomendações de péssima qualidade, já que o sistema não possui informações suficientes sobre os mesmos para lhes fazer recomendações personalizadas. Goldberg et al. (2001) trata o problema obrigando todos os usuários a inicialmente avaliar alguns itens previamente selecionados. Isto permite montar uma espécie de perfil para cada novo usuário. Porém, esta abordagem, apesar de eficiente, pode gerar uma rejeição ao uso do sistema, já que nem todos os usuários têm paciência para preencher este perfil.

Uma abordagem mais comum para tratar o problema consiste em recomendar os itens mais populares para os novos usuários (HERLOCKER et al., 2004), abrindo mão da característica de causar surpresa nestes casos.

2.4 Sistemas híbridos

Ao avaliar as características de SBCs e SFCs, conclui-se que ambos possuem vantagens e desvantagens.

Os SBCs possuem algoritmos dependentes do conteúdo, enquanto SFCs são independentes de conteúdo. Todavia, estes últimos possuem o problema do "início frio", enquanto os primeiros aliviam este problema ao obrigar o novo usuário a preencher um perfil. As matrizes de avaliação dos SFCs são muito esparsas, o que dificulta as análises.

Uma vantagem muito importante dos SFCs é que eles utilizam impressões de pessoas sobre os itens, tirando proveito da chamada inteligência coletiva. Além disso, esses sistemas podem gerar recomendações surpreendentes aos usuários, enquanto que os SBCs normalmente tendem a gerar recomendações óbvias e muito especializadas. Estas características estão resumidas na Tabela 2.

Baseados em conteúdo	***Filtragem Colaborativa***
Dependente de conteúdo	Independente de Conteúdo
Funciona com novos usuários	Não funciona com novos usuários
Não possui o problema da esparsidade	Problema da esparsidade
Não utiliza impressões humanas	Utiliza impressões humanas
Normalmente gera recomendações muito óbvias	Pode gerar recomendações surpreendentes

Tabela 2: Comparativo de sistemas baseados em conteúdo e de filtragem colaborativa

Nota-se que sistemas de filtragem colaborativa e os baseados em conteúdo possuem características complementares. Isto sugere que a junção das duas abordagens gere sistemas mais eficientes. De fato, alguns pesquisadores trabalham com sistemas híbridos.

Balabanovic e Shoham (1997) propõem o sistema de recomendação híbrido FAB, que indica páginas Web para seus usuários. As páginas são analisadas e seus termos mais significantes são extraídos. Feito isto, os usuários podem avaliar a página, indicando se gostaram ou não dela. A nota de cada usuário é então utilizada para aumentar ou diminuir o peso dos termos em seu perfil. Páginas cujos termos mais relevantes têm peso alto no perfil de um usuário são recomendadas a ele.

Além disso, o sistema FAB possui também características de filtragem colaborativa. Os perfis dos usuários, montados com técnicas baseadas em conteúdo, são comparados com outros perfis em busca de similaridades entre usuários. Estes perfis similares também são utilizados para prover recomendações a eles.

Já o sistema híbrido PTV, Personalized Television Listings, (SMYTH; COTTER, 2000) recomenda programas de televisão. Neste sistema as recomendações baseadas em conteúdo são extraídas através da análise de descrições textuais dos programas e as baseadas em filtragem colaborativa

utilizam as preferências dos usuários.

O algoritmo proposto neste trabalho é um algoritmo de filtragem colaborativa baseado em SVD e trata o problema da esparsidade da matriz de avaliações através da própria forma em que a SVD é executada, assunto que será abordado no Capítulo 4. Entretanto, o algoritmo não trata o problema do "início frio", sendo necessário extendê-lo com uma abordagem baseada em conteúdo para se melhorar as recomendações para novos usuários.

No próximo capítulo será apresentada com detalhes a Decomposição por Valor Singular, técnica utilizada pelo algoritmo proposto. Tal técnica é uma das formas de se realizar a Análise de Componentes Principais, que também será explorada no mesmo capítulo.

3 PCA E SVD

A moda é uma espécie de feiura tão intolerável que nós temos que alterá-la a cada seis meses.

— Oscar Wilde

O algoritmo proposto neste trabalho utiliza a técnica de fatoração de matrizes conhecida como Decomposição por Valor Singular (SVD). A SVD é uma das maneiras existentes para se resolver a Análise de Componentes Principais (PCA). Portanto, será abordado primeiramente a PCA, que é de fundamental importância para o entendimento da SVD.

3.1 Revisão conceitual

Este capítulo aborda conceitos básicos de álbegra linear, estatística e aprendizagem de máquina. Por essa razão será apresentado primeiramente uma pequena revisão dos conceitos mais importantes utilizados. Para um entendimento mais amplo destes conceitos, recomenda-se Boldrini et al. (1986), Barbetta, Reis e Bornia (2008) e Mitchell (1997), respectivamente. Todo o conteúdo desta seção foi baseado nestes três livros.

3.1.1 Estatística

A variância é uma medida estatística que indica o quão longe da média os valores de uma amostra estão. Ela é calculada através da média aritmética dos desvios quadráticos, sendo o desvio a diferença entre o valor real de uma amostra e sua média. Sendo n o tamanho da amostra, x_i o i-ésimo elemento e $\bar{x}$ a média, a variância (s^2) de uma amostra é dada por:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{3.1}$$

Já o desvio-padrão é bastante semelhante à variância, com a diferença que o desvio-padrão é representado na mesma unidade de medida da amostra.

Formalmente, o desvio-padrão é a raiz quadrada positiva da variância:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.2}$$

Outro conceito importante utilizado neste trabalho é o conceito de correlação. A correlação é uma associação numérica entre duas variáveis e representa o quanto as variáveis "caminham no mesmo sentido" ou "caminham em sentidos opostos". No primeiro caso, diz-se que as variáveis possuem correlação positiva e, no segundo, correlação negativa.

O coeficiente de correlação de Pearson é uma medida de correlação que não deve depender da unidade de medida em questão. Portanto é necessário fazer uma padronização dos dados antes de calcular a correlação. Sendo x_i o i-ésimo elemento de uma amostra, é necessário primeiramente obter cada x'_i, que respresenta o elemento x_i padronizado:

$$x'_i = \frac{x_i - \bar{x}}{s} \tag{3.3}$$

Com a padronização, variáveis correlacionadas positivamente tendem a possuir sinais iguais, tanto positivos quanto negativos. Já as variáveis correlacionadas negativamente tendem a possuir sinais diferentes.

Portanto, quando $\sum_{i=1}^{n} x'_i y'_i > 0$, diz-se que x está correlacionado positivamente com y, e quando $\sum_{i=1}^{n} x'_i y'_i < 0$, diz-se que x está correlacionado negativamente com y.

A correlação é medida através do coeficiente de correlação de Pearson (r), que varia entre -1 e 1. Quando r está próximo de 1, diz-se que a correlação é positivamente forte, e próximo de -1, negativamente forte. Formalmente, o coeficiente de correlação de Pearson pode ser definido como:

$$r(x,y) = \frac{\sum_{i=1}^{n}(x'_i y'_i)}{n-1} \tag{3.4}$$

O conceito de covariância é bastante similar ao conceito de correlação. A correlação entre duas variáveis é igual a covariância entre as variáveis divida pelo produto de seus desvios-padrão. Em outras palavras, a covariância

entre x e y é dada por:

$$cov(x,y) = r(x,y).s(x).s(y) \tag{3.5}$$

O valor esperado de uma variável aleatória discreta x, comumente defido por $E(x)$ e equivalente a média de x na amostra, é igual a soma da probabilidade ponderada dos possíveis valores de x. Dado Ω o espaço da amostra e $m(x)$ a função de distribuição da probabilidade de x, o valor esperado de x é dado por:

$$E(x) = \sum_{x \in \Omega} xm(x) \tag{3.6}$$

A covariância entre duas variáveis também pode ser expressa através do conceito de valor esperado:

$$cov(x,y) = E([x - E(x)][y - E(y)]) \tag{3.7}$$

Dado um vetor $\mathbf{X}$ contendo n variáveis aleatórias, a matriz de covariância Σ representa a covariância entre todas as variáveis deste vetor:

$$\Sigma = \begin{bmatrix} E([x_1 - E(x_1)][x_1 - E(x_1)]) & .. & E([x_1 - E(x_1)][x_n - E(x_n)]) \\ E([x_2 - E(x_2)][x_1 - E(x_1)]) & .. & E([x_2 - E(x_2)][x_n - E(x_n)]) \\ ... & .. & ... \\ E([x_{n-1} - E(x_{n-1})][x_1 - E(x_1)]) & .. & E([x_{n-1} - E(x_{n-1})][x_n - E(x_n)]) \\ E([x_n - E(x_n)][x_1 - E(x_1)]) & .. & E([x_n - E(x_n)][x_n - E(x_n)]) \end{bmatrix} \tag{3.8}$$

3.1.2 *Álgebra Linear*

Um espaço vetorial é um conjunto de vetores com algumas propriedades especiais, dado que a soma de quaisquer dois vetores deste espaço é igual a um outro vetor do espaço. Além disso, a multiplicação de qualquer vetor por um escalar também produz um vetor pertencente ao espaço.

As combinações lineares de vetores são operações de soma de vetores e/ou de produto dos mesmos por escalares. Tais operações geram sub-espaços vetoriais do espaço vetorial original, de modo que, com o menor sub-espaço, é possível obter qualquer outro vetor do espaço vetorial através de combinações lineares. Este menor sub-espaço vetorial linearmente independente é conhecido como base.

Sendo $\mathbf{V}$ um espaço vetorial qualquer contendo n vetores, diz-se que $\mathbf{V}$

é linearmente dependente se existe um vetor $\mathbf{v_i}$ que é uma combinação linear de outros vetores de $\mathbf{V}$. Dado a combinação linear $C(\mathbf{V})$ de todos os vetores de $\mathbf{V}$:

$$C(\mathbf{V}) = a_1\mathbf{v_1} + a_2\mathbf{v_2} + ... + a_n\mathbf{v_n} \tag{3.9}$$

Então, $\mathbf{V}$ é linearmente dependente se $C(\mathbf{V}) \neq 0$ e é linearmente independente de $C(\mathbf{V}) = 0$.

Dados dois espaços vetoriais $\mathbf{V}$ e $\mathbf{W}$, sendo $\mathbf{u}$ e $\mathbf{v}$ quaisquer vetores pertencentes a $\mathbf{V}$ e $k \in \mathbf{R}$, uma transformação linear $F : V \rightarrow W$ é uma função que satisfaz duas condições:

$$F(\mathbf{u} + \mathbf{v}) = F(\mathbf{u}) + F(\mathbf{v}) \tag{3.10}$$

$$F(k\mathbf{v}) = kF(\mathbf{v}) \tag{3.11}$$

Toda tranformação linear está asssociada a uma matriz. Para toda matriz $\mathbf{M}_{m\times n}$, existe uma transformação linear $T : \mathbf{R}^n \rightarrow \mathbf{R}^m$ equivalente. Desta forma, aplicar transformações lineares a vetores é equivalente à multiplição da matriz de transformação linear correspondente pelo vetor.

Sendo $\mathbf{v}$ um vetor de tamanho n e $\mathbf{M}_{m\times n}$ uma matriz de transformação linear, se a multiplicação da matriz pelo vetor não altera a direção do vetor, podendo possivelmente alterar seu tamanho ou sentido, diz-se que $\mathbf{v}$ é um autovetor da matriz $\mathbf{M}_{m\times n}$. Neste caso, λ é um valor escalar conhecido como autovalor da matriz e indica o quanto o vetor aumentou ou diminuiu de tamanho e se o mesmo mudou de sentido. Ou seja:

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v} \tag{3.12}$$

A relação de PCA com álgebra linear e estatística ocorre pois, para se fazer PCA, é necessário encontrar os autovetores e autovalores da matriz de covariância dos dados sendo analisados. Uma outra forma, por vezes mais eficiente de se fazer PCA, envolve processar a SVD da matriz de dados diretamente, estratégia utilizada pelo algoritmo proposto neste trabalho. As técnicas de PCA e SVD serão exploradas ainda neste capítulo.

3.1.3 Aprendizagem de Máquina

Aprendizagem de Máquina é uma disciplina que envolve muitas outras disciplinas, como álgebra linear, probabilidade e estatística, algoritmos,

banco de dados, inteligência artificial, teoria de complexidade computacional, entre outras. Ela tem por objetivo a construção de sistemas que, através da análise de dados empíricos existentes, evoluem automaticamente com a experiência adquirida.

O algoritmo de sistema de recomendação proposto neste trabalho é um bom exemplo de uso da aprendizagem de máquina, já que o mesmo aplica SVD nos dados de avaliações existentes para tentar melhorar as predições de avaliações inexistentes.

Para adquirir experiência e evoluir um sistema é necessário "treinar" o algoritmo. Este treinamento, ou aprendizagem, pode ser supervisionado ou não-supervisionado.

Treinamento supervisionado envolve a tarefa de treinar um algoritmo sabendo qual deveria ser a saída do mesmo. Desta forma pode-se controlar o erro produzido pelo algoritmo e ajustá-lo para diminuir o erro.

Ja no treinamento não-supervisionado, não se sabe qual deve ser a saída do algoritmo. Este tipo de treinamento normalmente envolve a análise dos dados para a detecção de padrões interessantes.

Para se treinar algoritmos é necessário determinar quanto tempo de treinamento será despendido. Este tempo é decorrente de cada iteração de treinamento, conhecida como época de treinamento, que normalmente é um parâmetro de tais algoritmos.

Outro parâmetro comum em algoritmos de aprendizagem de máquina é a taxa de aprendizado, que indica o quão rápido ocorre o treinamento. Entretanto, acelerar demais o treinamento pode comprometer a acurácia do resultado e este parâmetro deve ser escolhido com atenção.

Um método de treinamento bastante utilizado em aprendizagem de máquina é o gradiente-descendente. Tal método é utilizado no treinamento do algoritmo proposto neste trabalho e consiste em tentar encontrar um valor mínimo de uma função.

Normalmente tal função indica o erro obtido pelo algoritmo. Portanto, encontrar um valor mínimo para a função significa treinar o algoritmo para não errar muito e então "aprender" a resolver o problema em questão.

Para fazer isto, o método de gradiente-descendente calcula sucetivamente a derivada da função até que se obtenha um valor mínimo considerado satisfatório. Esta técnica pode ser melhor explicada na Figura 2, que mostra a aplicação do gradiente descendente para se encontrar o ponto mínimo de uma função parabolóide.

Entretanto, nem sempre a função a ser utilizada é tão simples como a apresentada na Figura 2. Muitas vezes tais funções possuem diversos "vales",

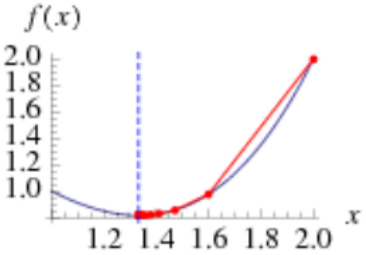

Figura 2: Exemplo da aplicação do gradiente descendente em uma função parabolóide

o que pode levar a técnica a encontrar mínimos locais ao invés do mínimo global. Esta é uma preocupação que deve ser levada em consideração, já que isto está diretamente relacionado com a qualidade de aprendizagem do algoritmo.

Outra questão muito importante que deve ser levada em consideração durante o treinamento de algoritmos é o fenômeno conhecido como *overfitting*. O *overfitting* pode ocorrer quando se treina demais um algoritmo de modo que o mesmo se especializa nos dados utilizados durante o treinamento e perde sua capacidade de generalização. No Capítulo 4 será apresentada a técnica utilizada neste trabalho, baseada no uso de coeficientes de regularização, que tenta evitar o problema de *overfitting* durante o treinamento.

O correto entendimento dos conceitos apresentados nesta seção é de fundamental importância para se compreender a técnica de PCA e SVD, que serão apresentadas na sequência, e o algoritmo proposto, apresentado no Capítulo 4.

3.2 Análise de Componentes Principais (PCA)

A análise de componentes principais (PCA) é uma técnica de análise estatística multivariada que explora a estrutura de variância-covariância de um conjunto de variáveis através de algumas combinações lineares destas variáveis (JOHNSON; WICHERN, 2007). A técnica foi primeiramente descrita por Karl Pearson (1901) e posteriormente aprimorada por Hotelling (1933), que descreveu o PCA de maneira passível de ser computada. Entretanto, somente depois do advento e da adoção em massa de computadores a técnica

passou a ser mais popular (MANLY, 1986).

PCA é utilizada em filtragem colaborativa para se encontrar padrões nos dados sendo analisados. Tais padrões podem ser interpretados como características dos itens e dos usuários dos sistemas. Deste modo, é possível associar itens com características mais próximas das características de um usuário e então recomendar estes itens a este usuário.

Matematicamente, o objetivo da PCA é utilizar p variáveis $\mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_p}$ e encontrar combinações lineares destas para produzir os componentes principais $\mathbf{Z_1}, \mathbf{Z_2}, ..., \mathbf{Z_p}$ que descrevem a variância dos dados. Cada componente $\mathbf{Z_i}$ têm a forma:

$$\mathbf{Z_i} = a_{i1}\mathbf{X_1} + a_{i2}\mathbf{X_2} + ... + a_{ip}\mathbf{X_p} \tag{3.13}$$

onde

$$a_{i1}^2 + a_{i2}^2 + ... + a_{ip}^2 = 1 \tag{3.14}$$

Os componentes são ordenados de forma decrescente conforme sua variância e, normalmente, apenas os primeiros componentes, com alta variância, são necessários para descrever o conjunto completo dos dados. Estes componentes também não estão correlacionados, o que indica que os mesmos podem representar diferentes dimensões dos dados. Desta forma, ao ignorar os componentes de baixa variância obtém-se uma redução dimensional dos dados iniciais.

O conceito de PCA pode ser melhor entendido através de um exemplo. A Figura 3 (STATCAN, 2007) mostra um gráfico contendo a estatura (altura) e a envergadura (maior distância entre as pontas dos dedos) de um grupo de homens. O eixo vertical indica a altura dos indivíduos e o eixo horizontal a envergadura.

Observa-se que existe uma correlação entre as duas variáveis, já que quanto mais alto é um homem, maior sua envergadura. Desta forma, pode-se obter uma reta que se aproxima dos dados apresentados e que poderia substituí-los, como ilustrado na Figura 3. Esta reta representa o componente principal com maior variância dos dados.

A PCA pode ser utilizada para se obter uma melhor visualização dos dados. Ao reduzir a dimensão de conjuntos de dados multidimensionais para duas ou três dimensões pode-se analisar e entender melhor os mesmos. Além disso, a PCA também é utilizada para fazer compressão de dados, já que a redução de dimensões economiza espaço.

Quando utilizada como pré-processamento de outros algoritmos de

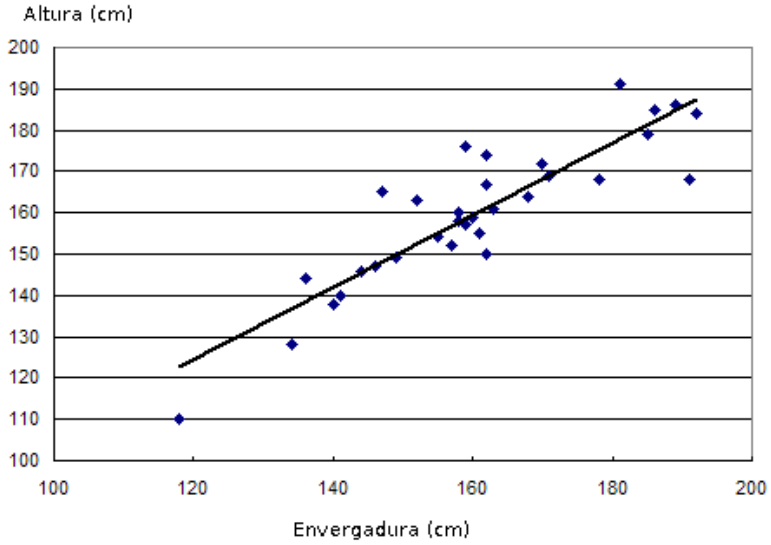

Figura 3: Gráfico que mostra a correlação positiva entre a estatura e a envergadura de um homem

aprendizado supervisionado, a redução dimensional da PCA diminui o tempo computacional e contribui para a redução de *overfitting*.

A PCA ainda pode ser utilizada para redução de ruído, pois instâncias com ruídos nos dados seriam ignoradas na redução dimensional. Outro uso comum de PCA ocorre no cálculo de distâncias, que por muitas vezes é mais fácil de fazer em dimensões reduzidas (JOLLIFFE, 2002).

A técnica PCA normalmente é utilizada em filtragem colaborativa para acelerar o treinamento. Goldberg et al. (2001) propõe o algoritmo Eigentaste que, a partir de um conjunto de avaliações comuns e obrigatórias feitas por todos os usuários, utiliza a redução dimensional para agrupar estes usuários e facilitar o processamento das recomendações. Por obrigar que todos os usuários avaliem um pequeno conjunto de itens, o algoritmo Eigentaste evita o problema da esparsidade, já que a matriz de avaliações é densa. Os autores alegam que o algoritmo possui tempo de processamento "online" constante (complexidade O(1)), pois grande parte do trabalho é feito antes de maneira "offline".

Honda et al. (2001) propõem uma forma diferente de utilizar PCA em filtragem colaborativa. O método utiliza a análise de componentes principais sob a matriz completa de avaliações, que normalmente é bastante esparsa. Deste modo, a redução dimensional gera uma matriz de rank menor que se aproxima da matriz de dados, ou seja, as predições de avaliações são realizadas utilizando-se uma aproximação da matriz inicial.

Para se encontrar os componentes principais de um conjunto de dados cuja escala não é uniforme, é necessário primeiramente fazer um pré-processamento de padronização. Dado um conjunto de dados qualquer representado por uma matriz $\mathbf{T_{m \times n}}$, sendo T_{ij} cada elemento deste conjunto, μ a média e σ o desvio-padrão, deve-se substituir cada elemento T_{ij} por $(T_{ij} - \mu)/\sigma$. Desta forma, a média do conjunto de dados é zerada e a variância se torna 1.

Feito o pré-processamento, pode-se extrair os componentes principais. Para facilitar o entendimento, analisaremos a redução dimensional de duas para uma dimensão. Neste caso, encontrar os componentes principais é equivalente a encontrar um vetor unitário **u** que maximiza a variância da projeção escalar dos dados neste vetor. Maximizar esta variância significa encontrar um vetor cuja somatória das distâncias dos pontos projetados em relação ao centro é maior. Este vetor será o que melhor representa os dados (NG, 2009).

A Figura 4 indica intuitivamente um vetor que representa bem os dados, ou seja, que poderia substituir os dados gerando um pequeno erro. Os dados são representados por X e as projeções por pontos. Observa-se que as projeções estão afastadas do centro cartesiano.

Já a Figura 5 indica um vetor que não representa bem os dados, ou seja, cuja substituição dos dados pelo mesmo gera erros maiores. Observa-se que, neste caso, as projeções dos dados estão próximas do centro cartesiano.

Portanto, quando as projeções dos dados em um vetor estão o mais longe possível do centro cartesiano, estas projeções possuem maior variância e, portanto, o vetor resultante é o componente principal **u** que se busca obter.

A intuição apresentada nas figuras anteriores pode ser explicada também matematicamente. A Figura 6 apresenta dois vetores **a** e **b** , sendo θ o ângulo entre eles. Tem-se que a projeção escalar de **a** em **b** é dada por **w**:

$$\mathbf{w} = |\mathbf{a}| \, cos\theta \tag{3.15}$$

Da álgebra linear, tem-se que:

$$\mathbf{a}^T \mathbf{b} = |\mathbf{b}| \, |\mathbf{a}| \, cos\theta = |\mathbf{b}| \, \mathbf{w} \tag{3.16}$$

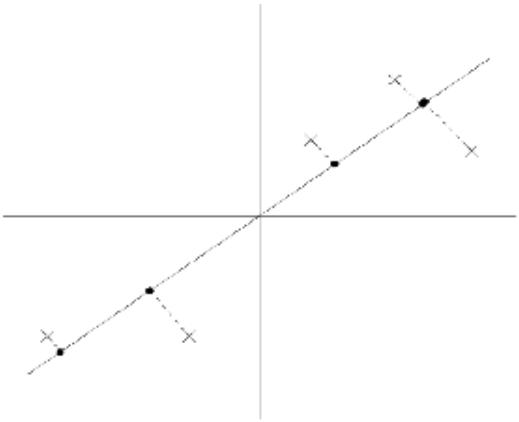

Figura 4: Exemplo de vetor que representa bem os dados

Ou seja, o produto escalar de **a** e **b** é igual ao produto do módulo de **b** com a projeção escalar de **a** em **b**. Substituindo **b** pelo vetor unitário **u** em 3.16, dado que $|\mathbf{u}| = 1$ já que **u** é um vetor unitário, obtém-se a projeção escalar de **a** em **u**:

$$\mathbf{a}^T\mathbf{b} = |\mathbf{a}|\,|\mathbf{u}|\,cos\theta = |\mathbf{a}|\,cos\theta \tag{3.17}$$

Ou seja, $\mathbf{a}^T\mathbf{u}$ é a projeção de **a** em **u**. Como dito anteriormente, para encontrar os componentes principais é necessário encontrar um vetor unitário **u** que maximiza a variância da projeção escalar dos dados sobre ele. Ou seja, é necessário encontrar um vetor que maximize o somatório das projeções dos dados em **u**. Sendo t o número de elementos do conjunto de dados, este somatório é aqui representado por:

$$\frac{1}{t}\sum_{i=1}^{t}(\mathbf{a}^T\mathbf{u})^2 \tag{3.18}$$

Desenvolvendo o somatório em 3.18:

$$\frac{1}{t}\sum_{i=1}^{t}(\mathbf{a}^T\mathbf{u})^2 = \frac{1}{t}\sum_{i=1}^{t}(\mathbf{a}^T\mathbf{u})(\mathbf{a}^T\mathbf{u}) = \frac{1}{t}\sum_{i=1}^{t}(\mathbf{u}^T\mathbf{a})(\mathbf{a}^T\mathbf{u}) = \mathbf{u}^T(\frac{1}{t}\sum_{i=1}^{t}\mathbf{a}\mathbf{a}^T)\mathbf{u} \tag{3.19}$$

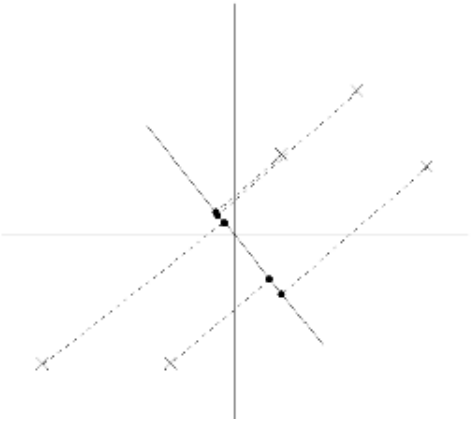

Figura 5: Exemplo de vetor que não representa bem os dados

Observa-se que $\frac{1}{t}\sum_{i=1}^{t}\mathbf{a}\mathbf{a}^T$ é a matriz de covariância, que será simplesmente chamada de Σ:

$$\mathbf{u}^T(\frac{1}{t}\sum_{i=1}^{t}\mathbf{a}\mathbf{a}^T)\mathbf{u} = \mathbf{u}^T(\Sigma)\mathbf{u} \tag{3.20}$$

Portanto, para encontrar o vetor $\mathbf{u}$ é necessário resolver o seguinte problema de otimização com restrição:

$$\arg\max_{\mathbf{u}} \mathbf{u}^T(\Sigma)\mathbf{u} \tag{3.21}$$

Existem diversas técnicas para se resolver tal problema, mas elas não serão abordadas neste trabalho. Para uma melhor compreensão, recomenda-se (GOLUB; Van Loan, 1996).

Ao se resolver este problema de otimização da PCA, $\mathbf{u}$ passa a ser o autovetor da matriz Σ. Generalizando para o caso de múltiplas dimensões, é preciso obter os autovetores com maiores autovalores da matriz de covariância.

Para resumir, o processamento da PCA se dá em três etapas:

1. Pré-processamento

2. Computar a matriz de covariância

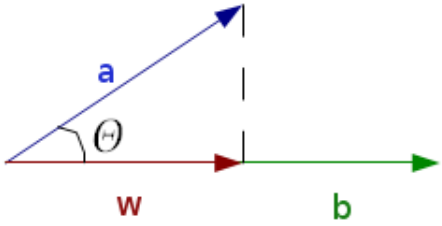

Figura 6: Projeção escalar (**w**) de **a** em **b**

3. Encontrar os autovetores com maiores autovalores da matriz

Entretanto, a matriz da etapa 2 pode ser enorme, o que torna a sua computação bastante difícil. Uma forma de se obter os autovetores de Σ sem precisar calcular a matriz Σ é através da Decomposição por Valor Singular (SVD), ou seja, o SVD pode ser utilizado para facilitar o processamento da PCA. É exatamente esta a relação que SVD tem com PCA. O próximo capítulo aborda o SVD com mais detalhes.

3.3 Decomposição por Valor Singular (SVD)

Decomposição por valor singular (SVD) é uma técnica algébrica de fatoração de matrizes, a qual pode ser usada para descobrir características latentes (i.e. escondidas) dos dados. SVD analisa qualquer matriz em busca de correlações e agrupa os dados correlacionados, causando uma redução dimensional na matriz. Dada uma matriz $\mathbf{T}_{n\times m}$, SVD a transforma no produto de três matrizes:

$$\mathbf{T}_{n\times m} = \mathbf{S}_{n\times n}\Sigma_{n\times m}\mathbf{V}^{\mathbf{T}}_{m\times m} \tag{3.22}$$

onde $\mathbf{S}$ é a matriz ortogonal cujas colunas são os vetores singulares da esquerda, $\mathbf{V^T}$ é a matriz ortogonal cujas colunas são os vetores singulares da direita, e Σ é a matriz diagonal de valores singulares positivos e decrescentes. É uma propriedade da SVD que as colunas da matriz $\mathbf{S}$ são os autovetores da matriz $\mathbf{T}\mathbf{T}^T$ e as colunas de $\mathbf{V}^T$ são os auto-vetores da matriz $\mathbf{T}^T\mathbf{T}$.

Portanto, ao aplicar o SVD em $\mathbf{T}$, obtém-se os auto-vetores da matriz $\mathbf{T}\mathbf{T}^T$, que é exatamente igual a Σ. Desta forma, o SVD pode ser utilizado

para se encontrar os auto-vetores da matriz de covariância sem precisar efetivamente calculá-la.

Além de ser utilizado para o processamento da PCA, o SVD também é comumente utilizado na área de processamento de sinais, reconhecimento de padrões, processamento de linguagem natural, entre outras áreas (JOHNSON; WICHERN, 2007).

O exemplo apresentado na Tabela 1, que contém avaliações de usuários a filmes, pode ser representado pela matriz **T**:

$$\mathbf{T} = \begin{bmatrix} 5 & 1 & 0 & 4 \\ 0 & 4 & 1 & 1 \\ 0 & 5 & 0 & 1 \\ 2 & 4 & 1 & 1 \end{bmatrix} \tag{3.23}$$

Ao aplicar o SVD nesta matriz, obtem-se a seguinte decomposição:

$$\mathbf{T} = \begin{bmatrix} -0,5 & 0,8 & -0,2 & -0,0 \\ -0,4 & -0,3 & 0,0 & -0,8 \\ -0,5 & -0,4 & -0,6 & 0,4 \\ -0,5 & -0,1 & 0,7 & 0,3 \end{bmatrix} \begin{bmatrix} 8,6 & 0 & 0 & 0 \\ 0 & 5,5 & 0 & 0 \\ 0 & 0 & 1,1 & 0 \\ 0 & 0 & 0 & 0,6 \end{bmatrix} \begin{bmatrix} -0,4 & 0,6 & 0,4 & 0,4 \\ -0,7 & -0,5 & -0,0 & 0,1 \\ -0,1 & -0,0 & 0,7 & -0,6 \\ -0,4 & 0,4 & -0,5 & -0,5 \end{bmatrix} \tag{3.24}$$

Ao observar a fatoração da matriz acima, a redução dimensional não fica clara, pois uma matriz $4_{\times}4$ foi obtida pelo produto de três matrizes: uma $4_{\times}4$ à esquerda, uma $4_{\times}4$ no centro e uma $4_{\times}4$ à direita. Entretanto, pode-se optar pelo uso de SVD truncado (XU, 1998), que representa apenas algumas dimensões, reduzindo o tamanho das matrizes de decomposição. Este procedimento implica na inserção de erro na fatoração. É uma propriedade do SVD que este erro introduzido é o menor erro quadrático médio (GORRELL, 2006).

Reduzindo para duas dimensões no exemplo anterior, tem-se:

$$\mathbf{T} \approx \begin{bmatrix} -0,5 & 0,8 \\ -0,4 & -0,3 \\ -0,5 & -0,4 \\ -0,5 & -0,1 \end{bmatrix} \begin{bmatrix} 8,6 & 0 \\ 0 & 5,5 \end{bmatrix} \begin{bmatrix} -0,4 & 0,6 & 0,4 & 0,4 \\ -0,7 & -0,5 & -0,0 & 0,1 \end{bmatrix} \tag{3.25}$$

Neste caso, fica evidente a redução dimensional nas matrizes de decomposição, já que as matrizes obtidas são $\mathbf{S}_{4\times2}$, $\Sigma_{2\times2}$ e $\mathbf{V}^T_{2\times4}$.

Uma maneira mais simples e comumente utilizada de fatoração SVD utiliza apenas duas matrizes de decomposição para aproximar a matriz inicial. Neste caso, pode-se, por exemplo, multiplicar a matriz diagonal Σ pela matriz da direita $\mathbf{V}^T$. No caso do exemplo anterior, obtém-se a seguinte fatoração:

$$\mathbf{T} \approx \begin{bmatrix} -0,5 & 0,8 \\ -0,4 & -0,3 \\ -0,5 & -0,4 \\ -0,5 & -0,1 \end{bmatrix} \begin{bmatrix} -3,4 & 5,1 & 3,4 & 3,4 \\ -3,8 & -2,7 & 0 & 0,5 \end{bmatrix} \tag{3.26}$$

Estas duas matrizes de decomposição obtidas representam as características interessantes descobertas de usuários e filmes, respectivamente. Mais detalhes sobre a relação de SVD com filtragem colaborativa serão apresentados na próxima seção.

3.4 SVD e Filtragem Colaborativa

SVD é uma das formas de fatoração de matrizes usadas em filtragem colaborativa (ZHANG et al., 2005a). Por exemplo, ao aplicar SVD a uma matriz de avaliações de filmes por usuários pode-se descobrir algumas características que fazem os usuários avaliar positiva ou negativamente certos tipos de filmes. Estas características podem ser gêneros de filmes, participação de atores famosos, quantidade de prêmios recebidos, ou mesmo características mais complexas que não se pode entender, mas que estatisticamente fazem sentido. Dado que $\mathbf{T}_{n\times m}$ é uma matriz que contém as notas atribuídas por usuários a itens, sendo que as linhas representam os usuários e as colunas os ítens, ao aplicar o SVD à matriz, tem-se:

$$\mathbf{T}_{n\times m} \approx \mathbf{U}_{n\times d}\mathbf{M}^T_{m\times d} = \mathbf{Q}_{n\times m} \tag{3.27}$$

onde **U** é a matriz formada por d vetores de características de n usuários, **M** é a matriz formada por d vetores de características de m filmes e **Q** é uma aproximação de **T**. Cada vetor-linha $\mathbf{U}_i \in \mathbf{U}(i < n)$ representa a relação do usuário i com as características descobertas e cada vetor-linha $\mathbf{M}_j \in \mathbf{M}(j < m)$ representa a relação do filme j com essas características.

Por exemplo, ao executar o SVD sobre a matriz de avaliação representada pela Tabela 1, descobriu-se duas características importantes: $C1$ e $C2$ (significando "filmes de ação" e "filmes românticos", por exemplo). Note que este é apenas um exemplo didático para facilitar o entendimento, já que não é possível interpretar o significado de cada característica.

Observa-se em 3.26 que os vetores obtidos pelo SVD, do usuário "Lais" e do filme "O Poderoso Chefão" são, respectivamente:

$\mathbf{U_L} = \begin{bmatrix} -0,4 & -0,3 \end{bmatrix}$
$\mathbf{M_P} = \begin{bmatrix} -3,4 & -3,8 \end{bmatrix}$

Estes vetores indicam que a característica $C2$ é a mais importante ao usuário "Lais", seguido pela característica $C1$, já que o valor de $C2$ no vetor do usuário é maior que o valor de $C1$ $(-0,3 > -0,4)$. Ou seja, pode-se interpretar que o usuário "Lais" gosta mais de filmes românticos do que de filmes de ação. Ainda, segundo este exemplo, o filme "O Poderoso Chefão" é um filme que tem mais ação do que romance, já que $-3,4 > -3,8$.

Uma maneira de obter a predição da nota que "Lais" daria ao filme "O Poderoso Chefão" seria através do produto escalar de um vetor de características pelo outro:

$\mathbf{U_L} \times \mathbf{M_P}^T = 2,5$

Como a nota predita é baixa, considerando uma escala de 1 a 5, conclui-se que o usuário não gostaria do filme em questão. Ao observarmos novamente a Tabela 1, nota-se que a nota predita para o usuário "Lais" é próxima da nota existente do usuário "Fátima" para o mesmo filme e que ambos usuários constumam atribuir notas de maneira parecida. Este foi um padrão encontrado pelo SVD.

No exemplo anterior as avaliações variam de 1 a 5 e a multiplicação dos vetores pode gerar uma predição final de nota fora deste intervalo. Para evitar este problema, é preciso normalizar os vetores antes do produto escalar. Uma solução alternativa seria simplesmente alterar as predições que estão abaixo da nota mínima para 1 e as que estão acima da nota máxima para 5.

Neste capítulo foi apresentado a fundamentação teórica necessária para o entendimento do algoritmo proposto baseado em SVD. Tal algoritmo será apresentado no próximo capítulo.

4 UM ALGORITMO DE FILTRAGEM COLABORATIVA BASEADO EM SVD

A imaginação governa o mundo.

— Napoleão

Sistemas de recomendação tem como objetivo encontrar itens que os usuários não conhecem e que sejam mais relevantes aos mesmos. Portanto, dada uma matriz $\mathbf{T}_{n\times m}$ de avaliações, normalmente bastante esparsa visto que os usuários não avaliam todos os itens disponíveis, o objetivo de sistemas de recomendação é preencher esta matriz com predições de avaliações que estes usuários dariam aos itens. Desta forma, os itens mais relevantes aos usuários seriam aqueles que possuem maiores notas preditas.

O algoritmo proposto neste trabalho usa a técnica de fatoração de matrizes SVD para resolver PCA de uma maneira eficiente e preencher uma matriz esparsa com as predições de notas que usuários dariam aos itens.

De maneira geral, o algoritmo analisa uma matriz de avaliações $\mathbf{T}_{n\times m}$ e fatora a mesma em um produto de duas outras matrizes com dimensão reduzida. O resultado deste produto é a matriz de predição das notas de usuários a itens.

Existem muitas formas de se processar a SVD (GOLUB; Van Loan, 1996). Optou-se, neste trabalho, por fazer a fatoração através do cálculo do gradiente-descendente de uma função de erro, técnica muito utilizada em treinamento de redes neurais *feed forward* (RUMELHART; HINTON; WILLIAMS, 1988).

4.1 Função de erro

Sendo n o número de usuários e m o número de ítens do sistema, a função de erro utilizada no algoritmo proposto é definida pela seguinte equação (MA, 2008):

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(T_{ij} - p(\mathbf{U}_i, \mathbf{M}_j))^2 \tag{4.1}$$

onde:

- $\mathbf{T}_{ij}$: avaliação conhecida do usuário i para o item j;
- $\mathbf{I}_{ij}$: indica presença de avaliação, na qual $I_{ij} = 1$ se o usuário i avaliou o item j e $I_{ij} = 0$ se o usuário i não avaliou o item j;
- $\mathbf{U}_i$: Vetor de características do usuário i;
- $\mathbf{M}_j$: Vetor de características do ítem j;
- $p(\mathbf{U}_i, \mathbf{M}_j)$: função de predição da nota do usuário i para o item j;

$$p(\mathbf{U}_i, \mathbf{M}_j) = \begin{array}{ccc} 1, & se & \mathbf{U}_i\mathbf{M}_j < 1 \\ \mathbf{U}_i\mathbf{M}_j, & se & 1 \leq \mathbf{U}_i\mathbf{M}_j \leq 5 \\ 5, & se & \mathbf{U}_i\mathbf{M}_j > 5 \end{array} \tag{4.2}$$

Observa-se que a função de predição de nota utilizada acaba por limitar a avaliação no intervalo de 1 a 5.

Calculando-se os gradientes da função de erro através da derivada parcial de E com relação a $\mathbf{U}$ e $\mathbf{M}$, obtém-se:

$$\nabla\mathbf{U} = \frac{\partial E}{\partial \mathbf{U}} = \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(T_{ij} - p(\mathbf{U}_i, \mathbf{M}_j))\mathbf{M}_j$$

$$\nabla\mathbf{M} = \frac{\partial E}{\partial \mathbf{M}} = \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(T_{ij} - p(\mathbf{U}_i, \mathbf{M}_j))\mathbf{U}_j \tag{4.3}$$

4.2 Evitando *overfitting*

Ao implementar algoritmos baseados em um modelo é necessário avaliar a capacidade de generalização do mesmo. Se o modelo gerado se adequar demais aos dados de treinamento, pode estar ocorrendo um *overfitting* e a capacidade de generalização do algoritmo pode ficar prejudicada. Ou seja, o algoritmo possui ótima eficácia com os dados de treinamento, mas pode possuir péssima eficácia com novos dados.

Uma forma de se evitar overfitting é pelo uso de coeficientes de regularização. Tais coeficientes podem ser usados para introduzir um ruído na função a ser analisada, diminuindo as possibilidades do modelo se adequar demais aos dados de treinamento.

O algoritmo aqui proposto utiliza um coeficiente de regularização k, que introduz um ruído na função de erro que se deseja minimizar com o

gradiente-descendente (PATEREK, 2007):

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(T_{ij} - p(\mathbf{U}_i, \mathbf{M}_j))^2 + \frac{k}{2}\sum_{i=1}^{n}\|\mathbf{U}_i\|^2 + \frac{k}{2}\sum_{j=1}^{m}\|\mathbf{M}_i\|^2 \quad (4.4)$$

Desta forma, os gradientes calculados na verdade são:

$$\nabla\mathbf{U} = \frac{\partial E}{\partial \mathbf{U}} = \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(T_{ij} - p(\mathbf{U}_i, \mathbf{M}_j))\mathbf{M}_j - k\mathbf{U}_i \quad (4.5)$$

$$\nabla\mathbf{M} = \frac{\partial E}{\partial \mathbf{M}} = \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(T_{ij} - p(\mathbf{U}_i, \mathbf{M}_j))\mathbf{U}_j - k\mathbf{M}_j \quad (4.6)$$

4.3 Processando o SVD

Um método simples de treinamento é avaliar todos os dados, calcular os gradientes do erro e atualizar as matrizes de decomposição **U** e **M** de maneira iterativa. Entretanto, este método pode ocasionar tempo computacional muito elevado quando utilizado com grandes bases de dados.

Pode-se, então, atualizar as matrizes **U** e **M** sem avaliar todos os dados de treinamento. O algoritmo aqui proposto utiliza uma forma incremental de se realizar SVD, na qual cada característica é treinada separadamente e a atualização das matrizes de decomposição é feita após a análise de cada avaliação do conjunto de treinamento. Ou seja, para cada nota atribuída por um usuário a um item, calcula-se o gradiente e atualizam-se as matrizes **U** e **M**.

É importante notar que com esta estratégia incremental apenas o vetor do usuário e do item em questão são atualizados. Neste caso, os gradientes obtidos são ligeiramente diferentes dos apresentados em 4.5 e 4.6:

$$\nabla\mathbf{U_i} = \frac{\partial E_{ij}}{\partial \mathbf{U_i}} = I_{ij}(T_{ij} - p(\mathbf{U}_i, \mathbf{M}_j))\mathbf{M}_j - k\mathbf{U}_i \quad (4.7)$$

$$\nabla\mathbf{M_j} = \frac{\partial E_{ij}}{\partial \mathbf{M_j}} = I_{ij}(T_{ij} - p(\mathbf{U}_i, \mathbf{M}_j))\mathbf{U}_j - k\mathbf{M}_j \quad (4.8)$$

O algoritmo incremental proposto neste trabalho pode ser melhor entendido através de seu pseudo-código apresentado em Algoritmo 1.

A complexidade de tempo do algoritmo é $O(d * e * r)$.

Algoritmo 1 Algoritmo SVD

1: **Input:**(d, μ, e, t)
   d: Quantidade de características a serem descobertas
   μ: Taxa de aprendizado
   e: Número de épocas de treinamento
   t: Quantidade de avaliações conhecidas
2: **Inicializar:** (U, M)
3: **Algoritmo:**
   Para cada característica (do total de d características):
     Para cada época (do total de e épocas, ou enquanto a precisão estiver aumentando):
       Para cada avaliação T_{ij} conhecida (do total de t avaliações):
         Computar $\nabla \mathbf{U}_i$
         Computar $\nabla \mathbf{M}_j$
         $\mathbf{U}_i \leftarrow \mathbf{U}_i - \mu \nabla \mathbf{U}_i$
         $\mathbf{M}_j \leftarrow \mathbf{M}_j - \mu \nabla \mathbf{M}_j$
4: **Output:** Matrizes de características **U** e **M** treinadas

O método computa os gradientes durante certa quantidade de épocas ou enquanto a precisão continuar aumentando. Ou seja, ele pára assim que verificar que a eficácia do algoritmo, medida através do erro médio quadrático da predição, começa a piorar. Para calcular a eficácia do algoritmo, é necessário conhecer as notas que se quer predizer, ou seja, a interrompição do treinamento ocorre apenas durante o treinamento do algoritmo. Esta técnica é chamada de early stopping e é realizada para evitar overfitting. Entretanto, o early stopping pode também achar ótimos locais e prejudicar a eficácia do método (PRECHELT, 1998). De fato o algoritmo proposto neste trabalho encontra ótimos locais e a eficácia do mesmo está diretamente relacionada com a quantidade de ótimos locais evitados durante o treinamento.

4.4 Inicialização das matrizes de fatoração

No pseudo-código apresentado anteriormente foi mencionada a inicialização das matrizes **U** e **M**, porém não explicitou-se como esta inicialização ocorre. Contudo, a forma de se inicializar estas matrizes pode influenciar na velocidade de convergência do gradiente-descendente e, portanto, é importante analisar algumas alternativas.

Pode-se inicializar as matrizes **U** e **M** de maneira trivial, escolhendo

uma constante qualquer, como por exemplo, 0,1 (FUNK, 2006).

Outra forma de inicializar as matrizes é fazê-la de forma que o produto escalar das mesmas resulte na média de todas as avaliações. Ou seja, supondo-se que a média de notas dos usuários para os itens do sistema seja $\bar{y}$, tem-se:

$$\mathbf{U}.\mathbf{M} = \begin{bmatrix} \bar{y} & \dots & \bar{y} \\ \dots & \dots & \dots \\ \bar{y} & \dots & \bar{y} \end{bmatrix} \tag{4.9}$$

Seja d o número de dimensões, U_{ij} e M_{ij} valores iniciais das matrizes. Para se inicializar as matrizes de forma que a predição final do algoritmo seja igual à média de avaliações do sistema faz-se:

$$U_{ij} = M_{ij} = \sqrt{\frac{\bar{y}}{d}} \tag{4.10}$$

Entretanto, inicializar todos os valores das matrizes de maneira igual pode prejudicar a convergência. Portanto, é sugerido adicionar um pequeno ruído $n(s)$ à equação 4.10, sendo este ruído uma variável aleatória de distribuição uniforme $[-s,s]$, com s pequeno (MA, 2008). Assim:

$$U_{ij} = M_{ij} = \sqrt{\frac{\bar{y}}{d}} + n(s) \tag{4.11}$$

Certamente existem alguns itens, assim como usuários, que possuem uma nota média maior do que outros. Entretanto, a abordagem anterior trata cada item ou usuário de maneira igualitária, o que também reduz a velocidade de convergência.

Este trabalho apresenta uma proposta de inovação na inicialização das matrizes de fatoração ao inicializar seus valores de forma que os valores de predição levem em consideração o usuário e o item em questão. Desta forma, utiliza-se a média de cada usuário e de cada item ao invés da média geral de notas. Sendo $\bar{w}_i$ a média das notas atribuídas pelo usuário i e $\bar{z}_j$ a média das notas recebidas pelo item j, tem-se:

$$U_{ij} = \sqrt{\frac{\bar{w}_i}{d}} + n(r) \tag{4.12}$$

$$M_{ij} = \sqrt{\frac{\bar{z}_j}{d}} + n(r) \tag{4.13}$$

Esta forma de inicialização acelera a converência do algoritmo, já que inicializa-se as matrizes com valores mais prováveis de representarem o gosto e as características dos usuários. Além disso, ao inicializarmos as matrizes desta maneira mais inteligente, o erro inicial do treinamento é menor e alguns mínimos locais são evitados. Desta forma, a eficácia do algoritmo melhora, já que o treinamento não pára nestes mínimos locais.

No próximo capítulo serão apresentados os experimentos realizados com o algoritmo proposto.

5 EXPERIMENTOS

A teoria é algo que ninguém acredita, exceto a pessoa que a criou. Um experimento é algo que todos acreditam, exceto a pessoa que o fez.

— Albert Einstein

Os experimentos com o algoritmo SVD proposto foram realizados no contexto do Netflix Prize, que oferece, além de uma grande base de dados, uma sistemática de avaliação bem definida.

Além dos experimentos realizados com o algoritmo proposto neste trabalho, serão apresentados, também neste capítulo, alguns detalhes com relação ao concurso e um outro algoritmo utilizado como comparação.

5.1 Netflix Prize

A Netflix é atualmente a maior locadora de filmes do mundo, possuindo mais de cem mil títulos e oito milhões de usuários nos EUA (BERNARTT, 2008). A empresa não possui lojas físicas e atua somente pela Internet, onde os usuários podem escolher os filmes que mais lhes interessam e receber em casa o DVD ou até mesmo baixá-lo pelo computador (NETFLIX, 2009).

No site da empresa, os usuários podem avaliar os filmes vistos com notas de 1 a 5. Estas avaliações são utilizadas pelo CineMatch, sistema de recomendação proprietário da empresa, para prover as recomendações.

Com o intuito de melhorar a precisão de seu sistema, a empresa criou em 2006 o concurso Netflix Prize (BENNETT et al., 2007). Este concurso tem como objetivo premiar com um milhão de dólares o algoritmo de sistema de recomendação que consiga atingir uma melhoria de 10% de precisão sobre o algoritmo proprietário da empresa. Para tanto, é fornecida aos competidores uma base de dados contendo um histórico de avaliações de seus clientes para servir como base de treinamento. Solicita-se de cada concorrente estimar a nota de um conjunto de pares usuário-filme, para os quais somente a Netflix conhece a nota atribuída. A equipe vencedora do concurso em Setembro de 2009 se chama BellKor's Pragmatic Chaos, que conseguiu atingir 10.06% de

melhoria sobre o CineMatch. A equipe foi formada pela junção de três outras equipes que contam com funcionários das empresas AT&T, Yahoo Labs, Commendo Research e pesquisadores independentes (NETFLIX, 2009).

A base de dados fornecida pelo Netflix Prize é usada neste trabalho para avaliar o algoritmo proposto. A base completa contém 100.480.507 avaliações de 480.189 usuários, referentes a 17.770 filmes. Esta base de dados pode ser representada por uma matriz, onde as linhas representam os usuários, as colunas representam os filmes e as células da matriz são as notas atribuídas pelos usuários aos filmes. Tal matriz possui cerca de 8,5 bilhões de células, sendo esta mais de 98% esparsa, isto é, só há avaliações para cerca de 2% dos pares usuário-filme que constituem células da matriz (100.480.507 / 8.532.958.530). Estes dados de treinamento são fornecidos na forma de arquivos texto (.txt) e, além deles, também são fornecidos mais dois arquivos: um de prova (probe.txt) e outro de classificação (qualifying.txt). O primeiro contém 1.408.395 notas para pares usuário-filme e deve ser utilizado para verificar a precisão do algoritmo durante o treinamento. O segundo arquivo, qualifying.txt, possui 2.817.131 pares usuário-filme para os quais somente a Netflix conhece as notas atribuídas. Deve-se então tentar predizer estas notas e enviar o resultado à Netflix, que calcula a precisão obtida e envia por e-mail o resultado.

Além destes dados, a Netflix também informa os títulos dos filmes e as datas das avaliações. Entretanto, o algoritmo proposto neste trabalho não utiliza estas informações.

A avaliação da precisão dos algoritmos concorrentes ao Netflix Prize é realizada através da raiz quadrada do erro médio quadrático (*RMSE*).

Definição 5.1.1 *: Seja $p(U_i, M_j)$ a predição obtida pelo sistema da nota de um usuário U_i para o item M_j, $S_{Ui,Mj}$ a nota verdadeira atribuída pelo usuário para o mesmo filme e n o número de predições realizadas. O RMSE é definido por:*

$$RMSE = \sqrt{\frac{\sum_{x=0}^{n} (S_{Ui,Mj} - p(U_i, M_j))^2}{n}} \tag{5.1}$$

Quanto menor o RMSE obtido por um algoritmo, maior a precisão do mesmo, já que esta medida é uma medida de erro. O sistema proprietário Cinematch da Netflix obteve um RMSE de 0,9525. Tal sistema utiliza o algoritmo K-Nearest Neighbors (KNN), bastante popular em filtragem colaborativa, que se baseia na semelhança entre usuários vizinhos.

5.2 Algoritmo KNN

O algoritmo K Nearest Neighbors (KNN) é um método de análise de vizinhança que utiliza uma técnica de aprendizado supervisionado não-paramétrico. O aprendizado é supervisionado porque é feita uma análise sobre os dados de treinamento e o que se deseja aprender é conhecido. O KNN é considerado um algoritmo não-paramétrico porque não supõe qualquer distribuição de probabilidade. Neste caso, o número de parâmetros do algoritmo está associado aos dados e não a distribuição dos mesmos. Ou seja, o número de parâmetros cresce conforme se aumenta os dados de treinamento. Este aumento no número de parâmetros ocorre porque o KNN armazena as instâncias de treinamento na memória para utilizá-las posteriormente. Por este motivo, o algoritmo é classificado na área de aprendizagem estatística como memory-based ou instance-based.

A técnica é utilizada para fazer predições (classificações ou regressões) nas áreas de data mining, reconhecimento de padrões, processamento digital de imagens, entre outras. Na área de filtragem colaborativa foi uma das primeiras técnicas utilizadas (BRADLEY; RAFTER; SMYTH, 2000). A idéia geral é descobrir usuários parecidos com um determinado usuário e, então, recomendar itens que seus vizinhos gostam e que provavelmente ele também irá gostar. Quando utilizado para calcular a semelhança entre usuários para fazer a recomendação, o KNN é chamado de user-based. Resnick et al. (1994) propõem um sistema de filtragem colaborativa que recomenda artigos *netnews* baseado na semelhança entre usuários, que é calculada através do Coeficiente de Correlação de Pearson (RODGERS; NICEWANDER, 1980) entre as avaliações dos usuários para os diversos artigos existentes.

As bases de dados utilizadas em filtragem colaborativa normalmente possuem muito mais usuários do que itens (SARWAR et al., 2001). Desta forma, a descoberta de usuários semelhantes se torna computacionalmente custosa, já que existe uma grande quantidade de usuários a se avaliar.

Para contornar este problema, Sarwar et al. (2001) propõem um algoritmo KNN que analisa a semelhança entre itens ao invés de usuários. Os autores demonstram que este tipo de abordagem é mais eficiente para o cálculo de similaridade, além de gerar recomendações com maior precisão.

O algoritmo de referência utilizado para comparar com o algoritmo proposto neste trabalho é um KNN item-based. Ele utiliza o Coeficiente de Correlação de Pearson para medir a similaridade entre itens e um tratamento de intervalos de confiança para controlar a precisão de cada estimativa dos

coeficientes de correlação. Tal algoritmo atingiu um RMSE no Netflix Prize de 0,9253, o que representa uma melhoria de 2,85% sob o algoritmo de referência do concurso (BERNARTT, 2008).

5.3 Implementação

Antes da implementação final do algoritmo SVD, foi desenvolvido um protótipo do mesmo utilizando a linguagem Python e a biblioteca NumPy (SCIPY, 2010), que oferece uma série de ferramentas para o desenvolvimento de software para computação científica. Este protótipo processa apenas um pequeno volume de dados e não era adequado para o processamento de um grande volume, pois o tempo de processamento e a memória utilizada pelo programa são muito grandes.

Optou-se pelo uso de C++ para a implementação final do algoritmo, já que esta linguagem permite o desenvolvimento de sistemas bastante eficientes. O compilador utilizado foi o g++ 4.4.1 da GNU e o sistema operacional no qual os experimentos foram realizados foi o Linux Ubuntu 9.10. Durante a implementação foram utilizadas algumas bibliotecas de apoio ao desenvolvimento contendo estruturas de dados básicas, rotinas de ordenação, etc. Não foi utilizada nenhuma biblioteca de cálculo numérico no programa desenvolvido. Todos os cálculos necessários pelo programa foram implementados em C++ dentro do mesmo.

Foi utilizado um computador com processador Intel Core 2 Duo de 2,80 GHz e 4GB de memória RAM. Para facilitar e acelerar o acesso aos dados foi utilizado uma técnica de mapeamento de arquivos em memória do Linux chamada de MMAP (OPENGROUP, 2010), em ambos os algoritmos.

Já que não se tinha acesso a um ambiente que permitisse o processamento eficiente de software multithread, o algoritmo foi desenvolvido de maneira single-thread, não possuindo nenhum tipo de paralelização, o que poderia diminuir o tempo de processamento do mesmo. Portanto, foi necessário um grande trabalho de otimização de processamento e memória do programa desenvolvido.

Além disso, o SVD foi treinado com os dados puros do concurso, não sendo realizado nenhum tipo de pré-processamento ou normalização.

5.4 Descrição dos experimentos

Os experimentos foram realizados no contexto do Netflix Prize, utilizando-se inicialmente o conjunto de prova (probe), cujas notas a serem

preditas são conhecidas. Desta forma, eram ajustados alguns parâmetros e quando um RMSE razoável era obtido, o algoritmo era executado com estes parâmetros ajustados no conjunto de classificação (qualifying), cujas notas são desconhecidas. Os ajustes dos parâmetros eram realizados um de cada vez, ou seja, os parâmetros dos algoritmos eram fixados e somente um era variado por vez. Assim pode-se manter um controle sobre a efetividade do ajuste de cada parâmetro.

O resultado do algoritmo, contendo as notas preditas, eram enviados à Netflix pela internet, que calculava o novo RMSE e informava a classificação do algoritmo no ranking do concurso.

O cálculo do RMSE realizado pela Netflix era elaborado para evitar fraudes. O conjunto de tuplas filme-usuário-nota enviados para classificação eram divididos em dois sub-conjuntos de forma aleatória, chamados de conjunto "quiz" e conjunto "test". O RMSE era calculado para o primeiro conjunto e a classificação do algoritmo era informada na internet. Por fazer esta divisão de forma aleatória, evita-se que alguém mal intencionado possa enviar um resultado, avaliar o RMSE, fazer uma pequena modificação no resultado, avaliar o RMSE novamente e então fazer inferências se a mudança realizada melhorou ou não o RMSE. Além disso, só era possível fazer uma submissão para classificação por dia.

Para avaliar o SVD com o conjunto probe foi necessário remover este conjunto dos dados de treinamento. Desta forma, a avaliação do RMSE com o conjunto de prova caracteriza a técnica de validação cruzada (KOHAVI, 1995), utilizada para se evitar o overfitting do modelo com os dados de treinamento. Nos experimentos realizados com o KNN não foi necessário fazer esta remoção, já que o KNN não é baseado em modelo mas sim em memória.

5.5 Resultados

Os parâmetros variados no SVD durante a experimentação foram: a quantidade de dimensões (d) a serem representadas e a quantidade de épocas de treinamento (e). Já no KNN, optou-se pela utilização do número de vizinhos (k) a serem analisados. As estatísticas analisadas dos experimentos foram o RMSE e o tempo de processamento.

Iniciou-se os experimentos com o algoritmo SVD utilizando 10 dimensões e 12 épocas. Este experimento gerou recomendações 1,47% piores que as do CineMatch. Optou-se então pelo ajuste do número de épocas. A Figura 7 mostra a relação do aumento do número de épocas com o aumento da precisão do algoritmo. Ao aumentarmos o número de épocas para mais de

120, notou-se que o RMSE estabilizou. Isto se deve ao fato de que grande parte das atualizações dos gradientes não atingiam o número de épocas estabelecido, pois o RMSE começava a aumentar e então o algoritmo interrompia o treinamento (early stopping). Portanto, fixou-se o número de épocas em 120 e novos experimentos com o aumento do número de dimensões foram iniciados.

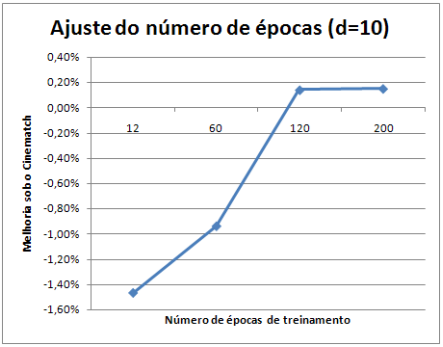

Figura 7: Ajuste do número de épocas (d=10)

A Figura 8 mostra o ajuste do número de características (ou dimensões) a ser buscadas pelo algoritmo. Ao aumentarmos o número de dimensões para 32, e posteriormente 64, o algoritmo passou a apresentar valores de RMSE mais baixos e melhorias mais significativas com relação ao algoritmo de referência da competição. O valor de 256 dimensões apresentou um resultado de RMSE muito bom, porém o tempo computacional aumentou bastante. Experimentos com mais de 256 dimensões não surtiram grandes efeitos. Portanto, estima-se que 256 dimensões são estatisticamente representativas no volume de dados utilizado nas experimentações. Em outras palavras, estima-se que existam 256 características que fazem com que grupos de usuários avaliem de forma parecida grupos de itens.

A Tabela 3 contém os resultados, ordenados de maneira decrescente de acordo com a melhoria sob o algoritmo referência, dos experimentos realizados com o algoritmo SVD.

Ao realizar experimentos com o algoritmo de comparação baseado em KNN, chega-se à conclusão de que o melhor algoritmo experimentado possui k igual a 23 e RMSE de 0,9253, o que representa 2,85% de melhoria sob o algoritmo de referência da competição (BERNARTT, 2008). A Tabela 4 contém

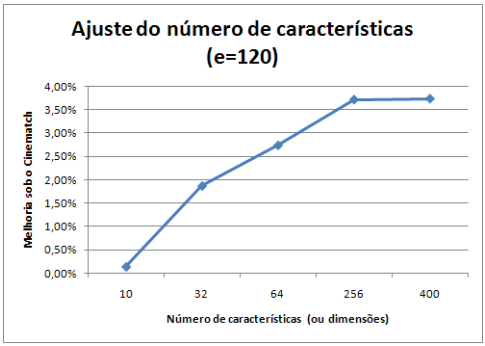

Figura 8: Ajuste do número de características (e=120)

os resultados dos experimentos realizados com o algoritmo KNN.

Portanto, o melhor resultado com o algoritmo KNN foi de 2,85% e o melhor resultado com o algoritmo SVD foi de 3,72%, uma diferença de 0,87% a mais para o algoritmo SVD.

Resolveu-se então juntar os dois algoritmos através de uma média ponderada do resultado de ambos. Ou seja, atribuiu-se um peso para o algoritmo SVD, outro peso para o algoritmo KNN e calculou-se uma média ponderada de cada predição. Foram realizados diversos experimentos com pesos diferentes e chegou-se ao melhor resultado de 60% para o SVD e 40% para o KNN. Desta forma, se o SVD prevê, por exemplo, a nota 2 e o KNN prevê a nota 4, a nota final predita seria de 2,8 ($0,6 \times 2 + 0,4 \times 4$).

Ao combinar os algoritmos através da média ponderada dos resultados obtidos pelo algoritmo SVD, representados pela Tabela 3, com os resultados obtidos pelo algoritmo KNN, representados pela Tabela 4, obteve-se os resultados apresentados na Tabela 5.

A melhor combinação do algoritmo KNN com o SVD alcançou o RMSE de 0,9016, o menor obtido nas experimentações e que representa uma melhoria de 5,34% sob o algoritmo de referência da competição. A Figura 9 mostra um gráfico comparativo entre todos os experimentos realizados com o algoritmo SVD, KNN e o algoritmo que combina SVD com KNN.

Tabela 3: Experimentos realizados com o algoritmo SVD

Algoritmo	***Parâmetros***	***Tempo***	***Probe RMSE***	***Qualifying RMSE***	***Melhoria sob referência***
SVD	d=400, e=120	32 horas	0,9220	0,9169	3.74%
SVD	d=256, e=120	18 horas	0,9223	0,9171	3,72%
SVD	d=64, e=120	48 minutos	0,9298	0,9264	2,74%
SVD	d=32, e=120	32 minutos	0,9423	0,9347	1,87%
SVD	d= 10, e=200	24 minutos	0,9599	0,9511	0,15%
SVD	d= 10, e=120	23 minutos	0,9601	0,9512	0,14%
SVD	d= 10, e=60	21 minutos	0,9682	0,9615	-0,94%
SVD	d= 10, e=12	15 minutos	0,9717	0,9665	-1,47%

Tabela 4: Experimentos realizados com o algoritmo KNN

Algoritmo	***Parâmetros***	***Tempo***	***Probe RMSE***	***Qualifying RMSE***	***Melhoria sob referência***
KNN	k=23	50 minutos	0,9073	0,9253	2,85%
KNN	k=25	50 minutos	0,9075	0,9258	2,80%
KNN	k=40	50 minutos	0,9073	0,9259	2,79%
KNN	k=30	50 minutos	0,9082	0,9260	2,78%
KNN	k=22	50 minutos	0,9073	0,9265	2,73%
KNN	k=20	50 minutos	0,9072	0,9270	2,67%
KNN	k=19	50 minutos	0,9073	0,9273	2,64%

5.6 Análise dos resultados

Os experimentos mostram que o algoritmo baseado em SVD, quando executado com um grande número de dimensões, produz resultados com menor RMSE do que o KNN. Entretanto, SVD com d grande necessita de muito tempo de processamento.

No caso do KNN, a variação do parâmetro K resultou em um impacto pequeno no RMSE obtido. Além disso, o tempo de processamento é praticamente constante devido à própria natureza do algoritmo.

Nota-se também que a combinação obtida através da média ponderada dos resultados do SVD com os resultados do KNN geraram uma melhoria

Tabela 5: Experimentos realizados com o algoritmo híbrido (KNN+SVD)

Algoritmo	***Parâmetros***	***Tempo***	***Probe RMSE***	***Qualifying RMSE***	***Melhoria***
SVD + KNN	d=256, e=120,k=23	18 horas	0,9029	0,9016	5,34%
SVD + KNN	d=400, e=120,k=25	32 horas	0,9033	0,9020	5,30%
SVD + KNN	d=64, e=120,k=40	50 minutos	0,9126	0,9113	4,32%
SVD + KNN	d=32, e=120,k=30	50 minutos	0,9200	0,9187	3,54%
SVD + KNN	d=10, e=120,k=22	50 minutos	0,9254	0,9241	2,98%
SVD + KNN	d=10, e=60,k=20	50 minutos	0,9355	0,9342	1,92%
SVD + KNN	d=10, e=12,k=19	50 minutos	0,9431	0,9418	1,12%

muito boa na precisão. Isto ocorre porque muitas vezes os padrões não identificados por um algoritmo são identificados pelo outro e vice-versa. Ou seja, os algoritmos provavelmente são complementares com relação a detecção de padrões.

Neste capítulo foi apresentado os experimentos realizados com o algoritmo SVD proposto, com um algoritmo KNN e com um algoritmo misto dos anteriores. A abordagem combinada das técnicas resultou em uma melhoria bastante significativa na precisão.

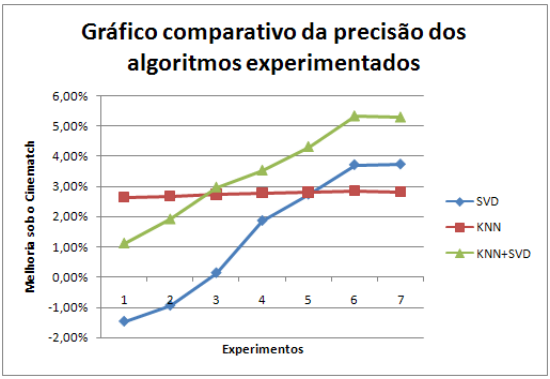

Figura 9: Gráfico comparativo da precisão dos algoritmos experimentados

6 CONCLUSÕES

Violência é o último refúgio do incompetente.

— Isaac Asimov

Este trabalho apresenta um algoritmo SVD para filtragem colaborativa que utiliza diversas abordagens presentes na literatura e algumas novas propostas. Tal algoritmo utiliza a técnica do gradiente-descendente para processar o SVD. A função de erro utilizada (MA, 2008) leva em consideração alguns coeficientes de regularização (PATEREK, 2007), que tentam evitar o *overfitting*.

O treinamento das características é feito individualmente e de maneira incremental. Ou seja, ao analisar cada avaliação do conjunto de treinamento, apenas os vetores de características do usuário e do item em questão são atualizados.

Este trabalho propõe uma nova forma de inicialização das matrizes de fatoração. Com esta inovação, a convergência do algoritmo ocorre mais rapidamente, além de melhorar a eficácia do mesmo. Os resultados preliminares deste trabalho foram publicados na Conferência Latino Americana de Informática 2009 (BOSCO et al., 2009).

Este algoritmo foi implementado e validado usando o benchmark da competição Netflix Prize, atingindo uma melhoria de 3,60% sobre o algoritmo referência do concurso. Participaram do concurso Netflix Prize 51.051 pesquisadores, 41.305 equipes de 186 países diferentes. Destas equipes, apenas 5.169 (12,51%) conseguiram uma submissão válida com RMSE menor que a referência do campeonato. Apenas 2,3% das equipes participantes conseguiram uma classificação melhor do que a obtida pelo algoritmo proposto neste trabalho.

O Netflix Prize foi o maior concurso já realizado na área de sistemas de recomendação e proporcionou um grande avanço desta área no mundo, com diversos artigos sobre o assunto sendo publicados e novas empresas na área de sistemas de recomendação sendo formadas.

O autor deste trabalho juntamente com o engenheiro de automação João Bernartt, autor do algoritmo KNN utilizado nos experimentos (BERNARTT, 2008), fundaram a Chaordic Systems S.A. (CHAORDIC, 2010), empresa focada em soluções de personalização em massa. O algoritmo apre-

sentado neste trabalho faz parte do conjunto de algoritmos utilizados pela empresa para prover recomendações a seus clientes.

O algoritmo também foi utilizado em conjunto com outro algoritmo da empresa no Netflix Prize e o melhor resultado obtido foi um RMSE de 0.9016, representando uma melhoria de 5.34% sob o algoritmo referência do concurso. Este resultado coloca o time na posição 398 do ranking, havendo apenas 0.9% de participantes com uma melhor colocação, como pode ser visto em destaque na Figura 10 (NETFLIX, 2010).

393	haha88	0.9014	5.36	2008-01-17 04:46:10
394	postechmlgport7	0.9014	5.36	2009-03-19 06:01:53
395	postechmlgport12	0.9014	5.36	2009-03-19 07:11:36
396	Nuts	0.9015	5.35	2007-12-10 09:04:08
397	Born in USSR	0.9016	5.34	2007-04-27 22:46:25
398	joaoboscoapf	0.9016	5.34	2009-04-16 22:58:49
399	QuemNaoTemColirio	0.9017	5.33	2009-04-01 20:53:39
400	Lambari	0.9017	5.33	2009-04-01 21:10:12
401	Chaordic Systems	0.9017	5.33	2009-04-16 21:50:51
402	jbapf	0.9017	5.33	2009-04-16 23:02:39
403	PLSA	0.9018	5.32	2007-02-09 16:22:30
404	neural nets	0.9018	5.32	2008-08-22 18:30:47

Figura 10: Ranking final do Netflix Prize

A aprendizagem teórica necessária para o desenvolvimento deste trabalho foi grande, já que não era um assunto dominado pelo autor desta dissertação. Entretanto, a maior dificuldade encontrada durante a elaboração deste trabalho foi a implementação do algoritmo. Desenvolver algoritmos de data mining que funcionam com um grande volume de dados não é algo trivial. É necessário fazer a análise assintótica de complexidade dos algoritmos, sempre equilibrando a relação do tempo de processamento com a memória utilizada pelo programa. Além disso, nem sempre é possível reutilizar bibliotecas de software já prontas, pois as mesmas não foram construídas para serem utilizadas neste contexto.

Os primeiros algoritmos desenvolvidos neste trabalho levariam anos para executar e foi necessário muito trabalho de otimização sobre eles para atingirmos o resultado apresentado no capítulo 5. A dificuldade de implementação deste tipo de algoritmo pode ser evidenciada pela quantidade

de submissões válidas enviadas ao Netflix Prize. Apenas 12.51% dos participantes conseguiram fazer uma submissão de um resultado qualquer ao concurso (NETFLIX, 2010).

Apesar de ter atingido uma boa colocação na competição, o algoritmo tem ainda muito o que melhorar.

6.1 Trabalhos futuros

O tempo de resposta do algoritmo é bastante elevado. A utilização de uma estratégia de paralelização do algoritmo pode melhorar muito a eficiência do mesmo.

Apesar de ter sido implementado para funcionar com diferentes bases de dados, o algoritmo só foi testado no contexto do Netflix Prize. Para se ter uma noção de quão genérico é o mesmo, é necessário testá-lo com outras bases de dados, possivelmente em diferentes domínios também.

O treinamento do algoritmo foi realizado sobre os dados de avaliações puras, não sendo utilizado qualquer tipo de pré-processamento destes dados. A aplicação de normalizações já provaram ser eficientes pré-processadores em filtragem colaborativa (BELL; KOREN, 2007). Com este tipo de normalização poderia-se, por exemplo, evitar efeitos estranhos que podem ocorrer com usuários muito rigorosos em suas avaliações.

Os experimentos também podem ser melhorados. A escolha de quais parâmetros devem ser ajustados foi realizada de forma empírica e sem um método bem definido. Técnicas de realização de experimentos podem ajudar na escolha dos melhores parâmetros (WU; HAMADA, 2009).

A forma de combinação do algoritmo com outros algoritmos também pode ser melhorada. A escolha dos pesos utilizados na média ponderada das predições das avaliações foi realizada de forma empírica. Um trabalho de otimização destes pesos poderia melhorar bastante o resultado dos algoritmos. Pode-se também utilizar outras estratégias de combinação dos algoritmos, como por exemplo, executar o SVD sobre os resíduos de um outro algoritmo, visando identificar padrões não identificados no primeiro.

Poderia-se também, em um trabalho futuro, combinar o algoritmo baseado em SVD proposto com uma outra abordagem baseada em conteúdo. Tal combinação seria interessante para combater o problema do "início-frio".

Uma outra possibilidade de melhoria da eficácia do algoritmo seria o uso de técnicas para se evitar mínimos-locais, como o uso da técnica de "têmpera simulada" (KIRKPATRICK, 1984), por exemplo.

6.2 Trabalhos relacionados

Existem alguns outros trabalhos que utilizam SVD com grandes bases de dados em filtragem colaborativa. Funk (2006) propõe um algoritmo baseado em SVD no qual as características são treinadas uma de cada vez. Além disso, em seu trabalho, a inicialização do modelo de treinamento é feita com valores fixos (0,1), o que certamente torna o treinamento um pouco demorado. Para evitar overfitting, é utilizado coeficientes de regularização. No algoritmo proposto neste trabalho, os coeficientes de regularização são divididos por dois para facilitar a derivada e no trabalho de Funk o coeficiente é usado diretamente.

O autor também faz experimentos com modelos não lineares. Usando uma função sigmóide, ele consegue modelar alguns padrões não lineares nas preferências dos usuários. Em suas experimentações no Netflix Prize, por exemplo, o autor descobre que predições de avaliações baixas são mais impactantes que previsões de avaliações altas e a função sigmóide utilizada pode modelar este padrão melhor que uma função linear.

Ma (2008) em sua tese de mestrado propõe o algoritmo CSVD (*Compound SVD*). Tal algoritmo utiliza uma função de erro com coeficientes de regularização exatamente iguais aos utilizados por este trabalho. Entretanto, o autor incorpora *bias* e restrições nos vetores de usuários e de itens. Esta estratégia mostrou-se eficaz nas bases de dados utilizadas durante as experimentações realizadas pelo autor. A inicialização do modelo de treinamento é feita de maneira similar à proposta neste trabalho. Entretanto, Ma Chih-Chao inicializa as matrizes de treinamento de usuários e de ítens de maneira igual, o que torna o tempo de treinamento necessário para se obter bons resultados maior que o tempo necessário pela abordagem apresentada neste trabalho.

O autor ainda compara diversas formas de treinamento do algoritmo proposto: O treinamento em batelada analisa todos os dados de treinamento para então calcular o gradiente e atualizar o modelo. Esta estratégia gera predições precisas, mas o tempo de processamento é proibitivo. A estratégia incremental incompleta atualiza o modelo assim que todas as avaliações de um usuário forem analisadas, o que acaba por acelerar o treinamento. Uma alternativa mais extrema é o treinamento incremental completo, que atualiza o modelo assim que cada avaliação for analisada. O autor conclui em seus experimentos que o treinamento incremental completo possui o maior custo-benefício.

Zhang et al. (2005b) utiliza SVD em filtragem colaborativa de uma

maneira diferente das apresentadas anteriormente. Ao autores utilizam a estratégia de *Expectation-Maximization* (EM) com SVD para se encontrar um modelo de baixa dimensionalidade que maximiza a função de log-verossimilhança (DEMPSTER; LAIRD; RUBIN, 1977) dos ratings em sistemas de recomendação.

EM é um algoritmo iterativo que consiste basicamente em duas etapas: *Expectation* e *Maximization*. Na primeira etapa, é feito uma estimativa da função de log-verossimilhança usando os parâmetros atuais do modelo de predição e na segunda etapa tais parâmetros são atualizados de maneira a maximizar a função encontrada na primeira etapa. Entretanto, ao usar esta estratégia, deve-se calcular o SVD para cada iteração do EM. Para reduzir o custo computacional requerido, os autores usam EM para reduzir a matriz de avaliações e então o SVD é aplicado apenas à esta matriz menor.

Nas duas primeiras abordagens apresentadas anteriormente, o problema da esparsidade é tratado utilizando-se um algoritmo de SVD incremental baseado no cálculo do gradiente-descentende de uma função de erro, ou seja, a esparsidade da matriz pode ser ignorada pois o algoritmo é incremental e em cada etapa de treinamento existe uma aproximação densa da matriz esparsa. A última abordagem apresentada utiliza EM para gerar uma matriz reduzida e completa. Neste caso, pode-se calcular o SVD através de técnicas convencionais, como o algoritmo Gram Schmidt (CH et al., 1992), por exemplo, já que o SVD será aplicado a matriz gerada pelo EM e esta não é esparsa.

REFERÊNCIAS

AMATRIAIN, X.; PUJOL, J.; OLIVER, N. I like it... i like it not: Evaluating user ratings noise in recommender systems. In: HOUBEN, G.-J. et al. (Ed.). *User Modeling, Adaptation, and Personalization*. [S.l.]: Springer Berlin / Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5535). p. 247–258.

ANDERSON, C. *The Long Tail: Why the Future of Business Is Selling Less of More*. [S.l.]: Hyperion, 2006. ISBN 1401302378.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *Modern Information Retrieval*. USA: Addison-Wesley Publishing Company, 2008. ISBN 0321416910, 9780321416919.

BALABANOVIC, M.; SHOHAM, Y. Fab: content-based, collaborative recommendation. *Commun. ACM*, ACM, New York, NY, USA, v. 40, n. 3, p. 66–72, 1997. ISSN 0001-0782.

BARBETTA, P. A.; REIS, M. M.; BORNIA, A. C. *Estatística para Cursos de Engenharia e Informática*. [S.l.]: Editora Atlas, 2008. ISBN 8522449899.

BELL, R.; KOREN, Y.; VOLINSKY, C. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2007. p. 95–104. ISBN 978-1-59593-609-7.

BELL, R. M.; KOREN, Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. *Data Mining, IEEE International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 43–52, 2007. ISSN 1550-4786.

BENNETT, J. et al. The netflix prize. kdd cup and workshop 2007. *SIGKDD Explor. Newsl.*, ACM, New York, NY, USA, v. 9, n. 2, p. 51–52, 2007. ISSN 1931-0145.

BERNARTT, J. L. V. *Um sistema de recomendação baseado em filtragem colaborativa*. 2008. UFSC.

BOLDRINI, J. L. et al. *Álgebra Linear*. [S.l.]: Editora Harbra, 1986. ISBN 8529402022.

BOSCO, J. et al. Explorando decomposição por valor singular em filtragem colaborativa. In: . [S.l.]: SBC, 2009.

BRADLEY, K.; RAFTER, R.; SMYTH, B. Case-based user profiling for content personalisation. In: *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. London, UK: Springer-Verlag, 2000. (AH '00), p. 62–72. ISBN 3-540-67910-3. Disponível em: <http://portal.acm.org/citation.cfm?id=647457-.727907>.

CH, S. M. et al. On the early history of the singular value decomposition. 1992. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc-/summary?doi=?doi=10.1.1.45.122>.

CHAORDIC. 2010. Http://www.chaordicsystems.com.

CHEN, H.-C.; CHEN, A. L. P. A music recommendation system based on music and user grouping. *Journal of Intelligent Information Systems*, Springer Netherlands, Volume 24, Numbers 2-3 / March, 2005, p. 113–132, 2005. ISSN 1573-7675.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, Blackwell Publishing for the Royal Statistical Society, v. 39, n. 1, p. 1–38, 1977. ISSN 00359246. Disponível em: <http://web.mit.edu/6.435/www/Dempster77.pdf>.

FUNK, S. *"Netflix update, Try this at home"*. December 2006. Http://sifter.org/simon/journal/20061211.html.

GOLDBERG, D. et al. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, ACM, New York, NY, USA, v. 35, n. 12, p. 61–70, 1992. ISSN 0001-0782.

GOLDBERG, K. et al. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 4, p. 133–151, July 2001. ISSN 1386-4564. Disponível em: <http://portal.acm.org/citation.cfm?id=593963.594023>.

GOLUB, G. H.; Van Loan, C. F. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)*. 3rd. ed. [S.l.]: The Johns Hopkins University Press, 1996. Paperback. ISBN 0801854148.

GORRELL, G. Generalized hebbian algorithm for incremental latent semantic analysis. In: *Proceedings of Interspeech*. [S.l.: s.n.], 2006.

HERLOCKER, J. L. et al. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, ACM, New York, NY, USA, v. 22, p. 5–53, January 2004. ISSN 1046-8188. Disponível em: <http://doi-.acm.org/10.1145/963770.963772>.

HONDA, K. et al. Collaborative filtering using principal component analysis and fuzzy clustering. In: *WI '01: Proceedings of the First Asia-Pacific Conference on Web Intelligence: Research and Development*. London, UK: Springer-Verlag, 2001. p. 394–402. ISBN 3-540-42730-9.

HU, Y.; KOREN, Y.; VOLINSKY, C. Collaborative filtering for implicit feedback datasets. In: *ICDM '08. Eighth IEEE International Conference on Data Mining, 2008.* [S.l.]: IEEE Computer Society, 2008. p. 263–272. ISSN 1550-4786.

JOHNSON, R. A.; WICHERN, D. W. *Applied Multivariate Statistical Analysis*. [S.l.]: Prentice Hall, 2007.

JOLLIFFE, I. T. *Principal Component Analysis*. Second. [S.l.]: Springer, 2002. Hardcover. ISBN 0387954422.

KIRKPATRICK, S. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, Springer Netherlands, v. 34, p. 975–986, 1984. ISSN 0022-4715. 10.1007/BF01009452. Disponível em: <http://dx.doi.org/10.1007/BF01009452>.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. p. 1137–1143. ISBN 1-55860-363-8.

MA, C. C. *A Guide to Singular Value Decomposition for Collaborative Filtering*. 2008.

MALTZ, D.; EHRLICH, K. Pointing the way: active collaborative filtering. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995. (CHI '95), p. 202–209. ISBN 0-201-84705-1. Disponível em: <http://dx.doi.org/10.1145/223904.223930>.

MANLY, B. F. *Multivariate statistical methods: a primer*. London, UK, UK: Chapman & Hall, Ltd., 1986. ISBN 0-412-28620-3.

MARSHAL, M. *"Aggregate Knowledge raises $5M from Kleiner, on a roll"*. 2006. Http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/.

MATLAB. 2010. Http://www.mathworks.com/products/matlab/.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997.

NETFLIX. 2006. Http://www.netflixprize.com/.

NETFLIX. 2009. Http://www.netflix.com/.

NETFLIX. 2010. Http://www.netflixprize.com/leaderboard?showtest=t&limit=1000.

NG, A. *"Professor Andrew Ng Homepage"*. 2009. Http://www-cs.stanford.edu/ ang.

OPENGROUP. 2010. Http://www.opengroup.org/onlinepubs/000095399/functions/mmap.html

PATEREK, A. *Improving regularized singular value decomposition for collaborative filtering*. 2007.

PEARL, P.; CHEN, L. User-involved preference elicitation for product search and recommender systems. *AI Magazine*, Association for the Advancement of Artificial Intelligence, v. 29, n. 4, p. 93–103, 2008. ISSN 0738-4602.

PRECHELT, L. "Early Stopping-But When?". In: *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS workshop*. London, UK: Springer-Verlag, 1998. p. 55–69. ISBN 3-540-65311-2.

R. 2010. Http://rss.acs.unt.edu/Rdoc/library/pcaMethods/html/00Index.html.

R. 2010. Http://www.r-project.org/.

RESNICK, P. et al. Grouplens: an open architecture for collaborative filtering of netnews. In: *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*. New York, NY, USA: ACM, 1994. p. 175–186. ISBN 0-89791-689-1.

RODGERS, J. L.; NICEWANDER, W. A. Thirteen Ways to Look at the Correlation Coefficient. *The American Statistician*, v. 42, n. 1, p. 59–66, 1980.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. MIT Press, Cambridge, MA, USA, p. 696–699, 1988.

SALTON, G.; BUCKLEY, C. *Term Weighting Approaches in Automatic Text Retrieval*. Ithaca, NY, USA, 1987.

SARWAR, B. et al. Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the 10th international conference on World Wide Web*. New York, NY, USA: ACM, 2001. (WWW '01), p. 285–295. ISBN 1-58113-348-0. Disponível em: <http://doi.acm.org/10-.1145/371920.372071>.

SCIPY. 2010. Http://numpy.scipy.org/.

SEGARAN, T. *Programming Collective Intelligence*. [S.l.]: O'Reilly Media, 2007. ISBN 0-596-52932-5.

SMYTH, B.; COTTER, P. *Personalized Electronic Program Guides for Digital TV*. 2000. Association for the Advancement of Artificial Intelligence (www.aaai.org).

STATCAN. 2007. Http://www.statcan.gc.ca/pub/12-593-x/2007001/figures/4183030-eng.htm.

TORRES, R. et al. Enhancing digital libraries with techlens+. In: *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*. New York, NY, USA: ACM, 2004. p. 228–236. ISBN 1-58113-832-6.

TORRES, R. D. *Combining Collaborative and Content-based Filtering to Recommend Research Papers*. 2004. UFRGS.

WU, C. F. J.; HAMADA, M. S. *Experiments : planning, analysis, and parameter design optimization*. [S.l.]: Lavoisier, 2009.

XU, P. Truncated svd methods for discrete linear ill-posed problems. *Geophysical Journal International*, Blackwell Publishing, Volume 135, Issue 2, p. 505–514, 1998.

YU, K. et al. Fast nonparametric matrix factorization for large-scale collaborative filtering. In: *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2009. p. 211–218. ISBN 978-1-60558-483-6.

ZHANG, S. et al. Using singular value decomposition approximation for collaborative filtering. In: *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*. Washington, DC, USA: IEEE Computer Society, 2005. p. 257–264. ISBN 0-7695-2277-7.

ZHANG, S. et al. Using singular value decomposition approximation for collaborative filtering. *E-Commerce Technology, IEEE International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 257–264, 2005. ISSN 1530-1354.